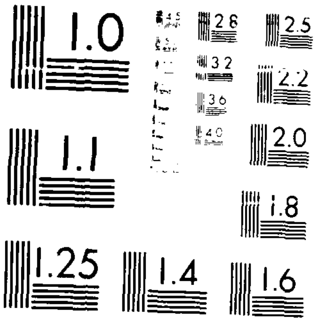
AD-A193 024 OPTICAL DATA PROCESSING(U) CARNEGIE-MELLON UNIV 1/2
PITTSBURGH PA DEPT OF ELECTRICAL AND COMPUTER
ENGINEERIN G D CASASENT APR 88 AFOSR-TR-88-0550
UNCLASSIFIED AFOSR-84-0293 F/G 12/9 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DTIC FILE COPY

AFOSR-TR- 88-0550

AD-A193 024

OPTICAL DATA PROCESSING

Annual Report on
Grant AFOSR-84-0293

SUBMITTED TO:

Air Force Office of Scientific Research
Bolling Air Force Base
Washington, D.C. 20332

ATTENTION: Dr. C. Lee Giles, AFOSR/NE

DTIC
ELECTE
MAY 1 9 1988
S
E

PREPARED BY:

David Casasent, Principal Investigator
Telephone: (412) 268-2464
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Period Covered: January 1987 - March 1988
Date of Report: April 1988

~~SECURITY CLASSIFICATION~~ OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S) AF-ODP-87

5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-0550

6a. NAME OF PERFORMING ORGANIZATION Carnegie Mellon University

6b. OFFICE SYMBOL *(If applicable)*

6c. ADDRESS *(City, State, and ZIP Code)*
Department of Electrical and Computer Engr.
Pittsburgh, PA 15213

7a. NAME OF MONITORING ORGANIZATION
AFOSR/NE
BLDG 410
BOLLING AFB, DC 20332-6448

8a. NAME OF FUNDING / SPONSORING ORGANIZATION AFOSR/NE

8b. OFFICE SYMBOL *(If applicable)* NE

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-84-0293

8c. ADDRESS *(City, State, and ZIP Code)*
Bolling Air Force Base
Washington, D.C. 20332
ATTENTION: Lee Giles

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION
61102F	2305	B1	

11. TITLE *(Include Security Classification)* Optical Data Processing

12. PERSONAL AUTHOR(S) David Casasent

13a. TYPE OF REPORT Annual

13b TIME COVERED FROM Jan.87 TO March88

14. DATE OF REPORT *(Year, Month, Day)* April 1988

15. PAGE COUNT 13

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Optical data processing, Pattern recognition, Feature extraction, Associative processors, Symbolic processors Rule based processors

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Our research concerns optical data processing for missile guidance and target recognition. It uses pattern recognition techniques with an increased use of knowledge base, inference machine and associative processor techniques. Our Year 3 work concerns new algorithms, real time and practical realizations of such systems, and new initial work on associative processors, symbolic rule-based processors and directed graph processors (with new attention to unique optical realizations of such systems).

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a NAME OF RESPONSIBLE INDIVIDUAL ~~David Casasent~~ Giles

22b TELEPHONE *(Include Area Code)* ~~(412) 268-2464~~ 202-767-4933

22c OFFICE SYMBOL NE

DD Form 1473, JUN 86

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PA
UNCLASSIFIED

TABLE OF CONTENTS

DTIC COPY INSPECTED 1

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification

By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

ABSTRACT

Our research concerns optical data processing for missile guidance and target recognitio It uses pattern recognition techniques with an increased use of knowledge base, inference machin and associative processor techniques. Our Year 3 work concerns new algorithms, real time an practical realizations of such systems, and new initial work on associative processors, symboli rule-based processors and directed graph processors (with new attention to unique optica realizations of such systems).

1. INTRODUCTION

The work in the past year of this grant (1 January 1987 - 31 December 1987) and its no-cost extension (January-March 1988) produced results on various new optical pattern recognition algorithms, real time laboratory results, new practical computer generated hologram recording techniques, and four new areas of potential work in optical artificial intelligence (these include associative processors, symbolic and rule based systems, as well as directed graph optical processors).

In this last year, the Principal Investigator (PI) and our AFOSR optical data processing effort were quite visible within the community. The PI served on the Defense Science Board Task Force on Image Recognition, gave 2 invited talks in non-optical processing conferences [1,2], an invited survey paper on optical pattern recognition and artificial intelligence [3], served on a NASA review committee on photonics, participated in several panel discussions, produced a book chapter on optical feature extraction [4], an encyclopedia article [7], plus numerous papers and conference presentations. This ends our pattern recognition AFOSR work. The results we have obtained should be of use in many future aspects of optical processing for image and scene analysis. These results are well-documented, due to our conscientious publication effort. These works have also been published in various non-optical journals to provide wider exposure for this technology.

We now highlight our research results in this third year of our work. Each result is more fully detailed in subsequent chapters, as noted. New pattern recognition algorithms and architectures devised included: new Hough transform techniques for distortion-invariant pattern recognition [5] were devised and demonstrated (Chapter 2 details these), a large 1000 class pattern recognition problem was addressed [6] with attractive initial results (Chapter 3 details this work), and a new string code processor [8] (detailed in Chapter 4) was advanced. Our second thrust area provided real time laboratory results of distortion-invariant pattern recognition using a liquid crystal television [9] (Chapter 5 details this work) and practical computer generated hologram (CGH) synthesis techniques using a laser printer were advanced [10] (Chapter 6 details this work). Our third major research area involved optical artificial intelligence processors. This work provided new results in associative processors, symbolic processors, rule based and directed graph processors. This included: new error correction associative processor concepts [11] as detailed in Chapter 7, new associative memory mapping realizations of an optical feature space [12] (Chapter 8), new heteroassociative memory processor performance measures and recollection vector encoding choices [13] (Chapter 9), symbolic and rule-based processors [14] as detailed in Chapter 10, and directed graph optical processor concepts and realizations [15] as detailed in Chapter 11. These last 5 items represent major new optical processing contributions to knowledge processing. Chapter 12 provides full documentation of our publications, presentations given, and theses produced related to this AFOSR effort. The 90 papers and over 100 technical talks presented in the three years of this program represent a quite major and significant contribution to optical data/information/knowledge processing research and to directions for future research in this area.

CHAPTER 1 REFERENCES

1. D. Casasent, "Optical Pattern Recognition and AI Algorithms and Architectures for ATR and Computer Vision", *Proc. SPIE*, Vol. 755, pp. 83-93, January 1987.

2. D. Casasent, "Electro Optic Target Detection and Object Recognition", *Proc. SPIE*, Vol. 762, pp. 104-125, January 1987.

3. D. Casasent, "Optical Pattern Recognition and Artificial Intelligence: A Review", *Proc. SPIE*, Vol. 754, pp. 2-11, January 1987.

4. "Optical Feature Extraction", D. Casasent, Chapter in Optical Signal Processing, pp. 75-95, Ed. by J.L. Horner, Pub. by Academic Press, San Diego, 1987.

5. R. Krishnapuram and D. Casasent, "Hough Space Transformations for Discrimination and Distortion Estimation", *Computer Vision, Graphics, and Image Processing*, Vol. 38, pp. 299-316, February 1987.

6. D. Casasent and A. Mahalanobis, "Optical Iconic Filters for Large Class Recognition", Applied Optics, Vol. 26, pp. 2266-2273, 1 June 1987.

7. D. Casasent, "Optical Information Processing", *Sixth Edition, Encyclopedia of Science and Technology*, McGraw-Hill.

8. D. Casasent and S.I. Chien, "Rule-Based String Code Processor", Proc. SPIE, Vol. 848, November 1987.

9. D. Casasent, S.F. Xia, A.J. Lee and J.Z. Song, "Real-Time Deformation Invariant Optical Pattern Recognition Using Coordinate Transformations", *Applied Optics*, Vol. 26, pp. 938-942, 15 March 1987.

10. A.J. Lee and D. Casasent, "Computer Generated Hologram Recording Using a Laser Printer", *Applied Optics*, Vol. 26, pp. 136-138, 1 January 1987.

11. S.A. Liebowitz and D. Casasent, "Error Correction Coding in an Associative Processor", Applied Optics, Vol. 26, pp. 999-1006, 15 March 1987.

12. R. Krishnapuram and D. Casasent, "Optical Associative Processor for General Linear Transformation", Applied Optics, Vol. 26, pp. 3641-3648, 1 September 1987.

13. D. Casasent and B. Telfer, "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results", Proc. SPIE, Vol. 848, November 1987.

14. D. Casasent and A. Mahalanobis, "Rule-Based Symbolic Processor for Object Recognition", Applied Optics, Vol. 26, pp. 4795-4802, 15 November 1987.

15. E. Baranoski and D. Casasent, "A Directed Graph Optical Processor", *Proc. SPIE*, Vol. 752, pp. 58-71, January 1987.

2. HOUGH TRANSFORM FOR PATTERN RECOGNITION

COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 38, 299-316 (1987)

Hough Space Transformations for Discrimination and Distortion Estimation

RAGHURAM KRISHNAPURAM AND DAVID CASASENT

Department of Electrical and Computer Engineering, Carnegie-Mellon University,
Pittsburgh, Pennsylvania 15213

Received November 12, 1985; accepted August 1, 1986

A new use of the Hough transform space defined for straight lines is described. The Hough space is used directly with new efficient distortion parameter transformations and template matching. This technique allows multiclass discrimination, intra-class distortion invariant recognition, and multiple distortion parameter estimation. A new hierarchical distortion parameter search method and spatial quantization in Hough space make realization of this technique very attractive. Performance of our algorithm on aircraft imagery and in the presence of noise is provided. © 1987 Academic Press, Inc.

1. INTRODUCTION

The Hough transform [1, 2], as suggested originally, is a method for detecting straight-line segments in an input image. This concept has been extended to include other analytically representable curves such as circles and ellipses [3]. It was further generalized to include arbitrary shapes and even three-dimensional (3-D) objects [4, 5]. These extensions are commonly referred to as generalized Hough transforms. The earlier versions of the generalized Hough transforms [6] required the computation of the gradient of each edge element and their storage in the form of a table. To reduce the computational burden, Davis [7] suggested a hierarchical Hough transform in which subpatterns of the image rather than the edge elements (pixels) were used as the basic units. The implementation of this approach is quite complex since we must deal with patterns rather than pixels.

Ballard and Sabbah [4] used a similar concept employing line segments rather than edge elements. They also suggested a different type of generalized Hough transform for detecting one type of object of arbitrary shape with scale, rotation, and translation differences present. They assume that the object boundary can be approximated by straight-line segments and that a list of the exact lengths, orientations and positions of all object boundary segments (with respect to a reference point on the object) is available. It is difficult but possible to obtain such a list for the model of the object being searched for. However, it is computationally burdensome to accurately obtain such a list for an input image, especially when noise is present. Implementing this efficiently will probably require a special symbolic language to handle the lists, especially when the lists are complicated. Detecting peaks in the Hough domain is difficult, especially when bias and noise are present [8]. It is difficult to quantify how well any such method will work when extraction of line segments in the input image is not easily achieved. The performance of such methods in the presence of noise is also not easily analyzed. All of these methods presuppose that the type of object being searched for is known in advance, i.e., they are only applicable to one-class problems and do not easily provide discrimination against other object types.

0734-189X/87 $3.00
Copyright © 1987 by Academic Press, Inc.
All rights of reproduction in any form reserved.

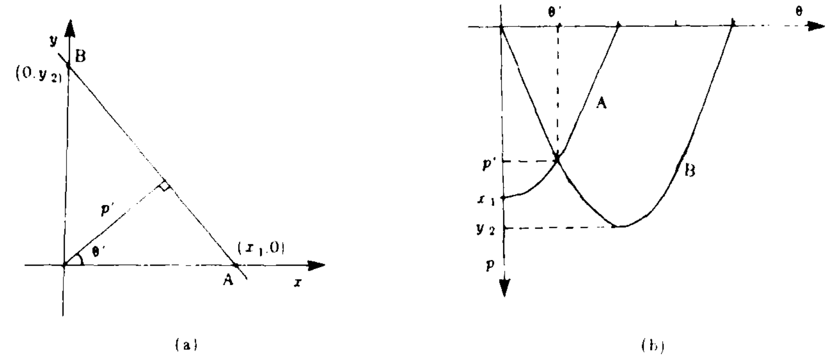


FIG. 1. Image plane to Hough transform plane mapping. (a) Points A and B in the image plane are mapped to (b) curves A and B in the Hough transform plane. The line in (a) maps to the point (p', θ') in (b).

The basic Hough transform for straight lines can be readily implemented digitally if the conventional parameterization in terms of the normal distance p and angle θ for straight lines is used. Figure 1 shows this classic image plane $f(x, y)$ to Hough transform plane $H(\theta, p)$ mapping for a line. Each point (x, y) in $f(x, y)$ is mapped to a sinusoidal curve in $H(\theta, p)$ given by

$$x \cos\theta + y \sin\theta = p. \tag{1}$$

This sinusoidal curve gives the p and θ parameters of all the straight lines passing through the point (x, y). Each point on the straight-line maps to a different sinusoidal curve (e.g., A and B in Fig. 1b) given by Eq. (1). All these curves intersect at a point in the Hough space and this point defines the p and θ parameters for the straight line shown in Fig. 1a.

The calculation of this Hough transform requires only simple multiplications involving trigonometric functions. Since the same multiplications are performed for every edge pixel in the image, computation of the Hough transform can be achieved in parallel. The results are accumulated in the $H(\theta, p)$ Hough array. It has also been shown [9] that the Hough transform is a special case of the Radon transform and that it can also easily be computed using optical techniques at video rates [10, 11]. This transform and $H(\theta, p)$ is thus very attractive for the low-level representation of images of objects.

This paper describes an alternative approach to estimation of the scale, rotation, and translation of an input image with respect to a reference image. It uses the basic straight-line Hough transform space. The proposed method is unique because it is capable of handling multiclass problems. Our approach is also original because the matching is performed directly in the Hough space. This differs significantly from the other approaches in which Hough techniques (i.e., accumulating votes in a 2-D or 3-D parameter array) are used for matching tables. In Section 2, we review the ease with which one can obtain the Hough transform of the input image. Section 2 also discusses the various applications and realizations of the Hough technique and the advantages and disadvantages of each. In Section 3 we detail our use of the Hough space for distortion invariance. This involves new transformations applied to the Hough space. The ease with which the transformations can be achieved is discussed. A hierarchical matching technique is detailed in Section 4 that significantly reduces the computations required to determine the object class and the object orientation. The image database used and the results obtained are then advanced (Sect. 5) and noise performance is also provided. Finally, Section 6 summarizes our work and advances our conclusions.

2. THE HOUGH DOMAIN AS A 2-D FEATURE SPACE

The algorithms suggested thus far to estimate the scale, rotation, and translation parameters of an input image using the Hough technique require the compilation of some form of a list or table. This list can be precomputed, as in [4], or dynamically computed, as in [5]. The *R*-table [6] requires the storage of a list of the gradients of all edge elements and their positions with respect to a reference point for the object to be searched for. For an unknown input image, the location of the reference point must be determined. To achieve this, an accumulator or Hough array is created with each element denoting a possible location of the reference point in the input image. The list from the model is used to compute the possible locations of the reference point with respect to each edge element in the input image, where each possible location corresponds to a particular translation and rotation of the object. Thus, each edge element in the input image votes for all possible locations of the reference point and these votes are accumulated in the Hough array. When the voting process has been completed for all edge elements, the peaks in the array indicate the possible locations of the reference point in the input image and thus denote the object's possible location. A similar approach using line segments rather than edge elements has been suggested [4].

In both cases, if the scale, rotation and translation parameters of the object are to be estimated simultaneously, a 4-D Hough array is needed. In this array, two dimensions denote the two translation parameters and the remaining two dimensions denote rotation and scale. This significantly increases the computational complexity and the memory requirements. Peak detection can be very difficult in such a 4-D array [8] since we must deal with hyper-surfaces. To overcome some of these problems, a two-level approach has been recommended in [4], in which the scale and orientation are estimated first (using a 2-D Hough array) and then translation is estimated (in a second-level 2-D Hough array). The digital implementation of these methods is straightforward and can be realized in parallel [12], given sufficient hardware and once the list has been obtained from the model and the line segment information has been extracted from the input image. (Accurate calculation of the line segment data from the input image can be very difficult.)

To reduce the memory requirements and computational burden, another approach has been suggested by Li, Levin, and LeMaster [13]. Here the voting process is carried out only in those parts of the Hough array where peaks are likely to occur. This method, however, applies only to situations in which an element in the input image votes on a hyperplane (and not on the more general hypersurface) in the parameter (Hough) space. It is also not known how well the method will perform when the peaks are diffused.

Several problems associated with these prior methods are worth noting. As Ballard and Sabbah point out [4], the position information is ignored while estimating the scale and orientation in the first level. As a result, peaks can occur in the accumulator array due to line segments in the input image that do not even touch each other and due to line segments that do not even lie near each other. Thus many false peaks can and do arise in the accumulator array. Another potential problem [8] with these methods is the detection of the peaks in the Hough array. Because of the inherent noise and bias present in the Hough transform, sharp peaks rarely occur, rather all peaks are distorted and diffused (smeared). Thus, we require the detection of local peaks rather than global peaks, and hence, sophisticated peak threshold methods. This problem becomes much worse when the dimensionality of the Hough array is large, since we must then deal with hypersurfaces. Another major problem with these prior methods is that they require the detection of the gradients and the positions of the edge elements in the input image, prior to the application of the Hough technique. If line segments are used, their orientations and positions are required. This image preprocessing often requires special edge-following and line-fitting algorithms which can be inaccurate and tedious. The final and quite a major problem with all of these methods is that they are object-specific and must thus be reformulated if a new object is to be searched for.

In this paper, we describe a different usage of the Hough transform to overcome these problems. In what follows, it should be understood that by Hough transform (HT), we mean the basic Hough transform defined for straight lines.

A simple and fast method of computing the lengths and orientations of the line segments in the input image is to use the Hough transform itself. The peaks in the Hough transform give the strengths and orientations of all lines in the input image. However, it suffers from the same problem as do the earlier methods since peak detection can again be difficult. Therefore, our new suggestion is not to extract any information from the Hough transform, but to simply use the Hough space as it is.

The basic idea of our approach is to approximate an object by a set of line segments and to describe these segments by a given 2-D pattern in the Hough domain. Thus, two similar objects would have similar Hough transforms and two different objects would have different Hough transforms. If the object is scaled, rotated, or translated, the Hough transform will change and distort. However, as we detail in Section 3, it is possible to define new transformations in the Hough domain that can remove these distortions and reconstruct the Hough transform of the original object in the reference orientation. When this is done, a simple template matching with the Hough transforms of different reference objects determines if the input object is a distorted version of a given object. It also determines the class of the object and its distortion parameters. This method can thus be used to discriminate between different types of objects (from the similarity of the template matches of their respective Hough transforms).

Eight distinct advantages of this approach are now noted. (1) It does not require extracting orientation and position information of edge elements or the lengths and orientations of line segments in the input image. (2) We do not need to detect the peaks in the Hough domain. The inherent Hough bias will reduce our discrimination capability, but it is not a serious problem unless the two objects are very similar. (3) This technique uses only a 2-D Hough space and thus there is no concern with hypersurfaces. As a result, (4) real-time computation is possible, and

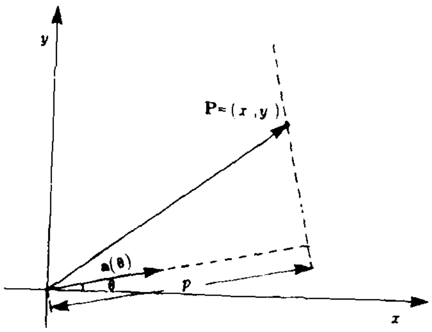


FIG. 2. Position vector **P** unit vector **a**(θ) and projection p as defined by Eqs (2) and (3)

(5) memory requirements are small. Memory requirements can be further reduced by coarsely discretizing the parameters of the Hough space. Because we use the Hough space itself, considerable quantization is allowed. (6) By using multiple Hough space reference patterns, this method can be used for multi-class problems. (7) The use of this Hough space as a 2-D pattern in a correlator is attractive and allows shift invariance. (8) Last, this approach can be easily extended to the recognition of 3-D range images and to the detection of 3-D orientation and translation. This can be achieved without increasing the dimensionality of the Hough space (as we will detail in a future publication).

3. HOUGH SPACE DISTORTION TRANSFORMATIONS

In this section, we present a vector description of the Hough transform for distorted objects. Our Hough space distortion transforms then directly follow.

3.1. *Vector Description*

In this approach, each point (x, y) in the image is represented by a position vector $\mathbf{P} = x\mathbf{i} + y\mathbf{j}$ from the origin as shown in Fig. 2. Here **i** and **j** are unit vectors along the x and y directions, respectively. The point **P** shown can lie on many (theoretically an infinite number of) lines that pass through it. Each of these straight lines can be characterized by a unit vector $\mathbf{a}(\theta)$ and a magnitude p. The unit vector $\mathbf{a}(\theta)$ extends from the origin perpendicular to the line and at an angle θ with respect to the positive x axis and p is the shortest projection distance from the origin to the line. The unit vector is described by

$$\mathbf{a}(\theta) = (\cos\theta)\mathbf{i} + (\sin\theta)\mathbf{j} \tag{2}$$

and the projection is defined by

$$\mathbf{P} \cdot \mathbf{a}(\theta) = p. \tag{3}$$

By varying θ and performing the required vector inner products in (3), we can easily generate the $\mathbf{a}(\theta)$ vectors and the corresponding p values for all possible straight lines passing through a particular point **P**.

We consider only a finite number of θ values between 0 and 2π. Thus, the process described above generates a finite list of (θ, p) pairs that characterize the corresponding straight lines passing through **P**. The point **P** is said to "vote" for all of those (θ, p) pairs in the Hough space. To represent the Hough space as a finite 2-D array, we discretize the values of p also. When the votes for all (θ, p) pairs have been accumulated for all points or edge elements in the input image, then the result is the discrete Hough transform of the input image. We assume that p is positive. If $\mathbf{P} \cdot \mathbf{a}(\theta) < 0$, we ignore the corresponding (θ, p) vote, since this implies $\mathbf{P} \cdot \mathbf{a}(\theta + \pi) > 0$ and that the associated vote would be counted at $(\theta + \pi, p)$. If $p = 0$, this corresponds to a line through the origin and for this case, (θ, p) and $(\theta + \pi, p)$ represent the same straight line. Thus, we need consider θ values only between 0 and π for the top $p = 0$ row in our plots and computations.

3.2. Hough Transform of a Scaled Image

Let $I_s(x, y)$ be a scaled version of $I(x, y)$ with scale factor s, such that a point **P** at (x, y) maps to a point $\mathbf{P}_s$ at $(x/s, y/s)$. Since $\mathbf{P}_s \cdot \mathbf{a}(\theta) = p/s$, the votes that occurred at (θ, p) in the original Hough transform now occur at $(\theta, p/s)$ for this scaled object. Thus, the Hough transform is compressed or expanded along the p axis only, depending on whether $s > 1$ or $s < 1$. The Hough transform $H_s(\theta, p)$ of the scaled image $I_s(x, y)$ is thus related to the Hough transform $H(\theta, p)$ of the original image by

$$H_s(\theta, p/s) = H(\theta, p). \tag{4}$$

The above equation can thus be used to reconstruct $H(\theta, p)$ from $H_s(\theta, p)$ as we detail later.

3.3. Hough Transform of a Rotated Image

Let $I_r(x, y)$ be the original image rotated in the image plane by an angle ϕ. In Fig. 3, we show one point **P** on the original object and the associated point $\mathbf{P}_r$ on the rotated object. In polar coordinates, **P** lies at (r, ψ) and $\mathbf{P}_r$ lies at $(r, \psi + \phi)$. Since

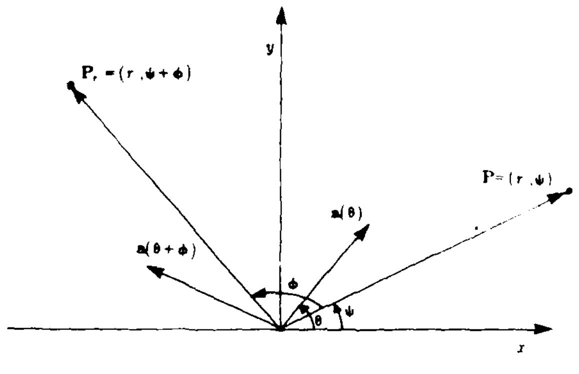


FIG. 3. A point **P** on the object, and its position ($\mathbf{P}_r$) when the object is rotated about the origin by an angle ϕ.

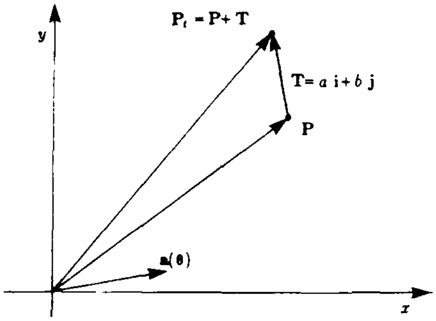


FIG. 4. A point **P** on the object, and its position $\mathbf{P}_t$ when the object is translated by **T**

$\mathbf{P} \cdot \mathbf{a}(\theta) = p = \mathbf{P}_r \cdot \mathbf{a}(\theta + \phi)$, it follows that the votes at (θ, p) in the original $H(\theta, p)$ now occur at $(\theta + \phi, p)$ in the Hough transform $H_r(\theta, p)$ of $I_r(x, y)$. The new and original transforms are thus related by

$$H_r(\theta + \phi, p) = H(\theta, p). \tag{5}$$

To obtain the original Hough transform from $H_r(\theta, p)$ of the rotated image, we need merely shift the Hough array horizontally by an amount equal to the rotation ϕ. This shift is a circular shift since the points (θ, p) and $(\theta + 2\pi, p)$ are equivalent in the Hough domain.

3.4. Hough Transform of a Translated Image

Let $I_t(x, y)$ be the image obtained by translating the object by (a, b) and let $H_t(\theta, p)$ be its Hough transform. A point **P** in the original image will now lie at $\mathbf{P}_t = \mathbf{P} + \mathbf{T}$, where the translation vector $\mathbf{T} = a\mathbf{i} + b\mathbf{j}$ is shown in Fig. 4. We let the projection magnitude be $\mathbf{P} \cdot \mathbf{a}(\theta) = p$ for a line corresponding to an angle θ. Then, the projection magnitude for the translated point is computed as

$$\begin{aligned} \mathbf{P}_t \cdot \mathbf{a}(\theta) &= (\mathbf{P} + \mathbf{T}) \cdot \mathbf{a}(\theta) = \mathbf{P} \cdot \mathbf{a}(\theta) + \mathbf{T} \cdot \mathbf{a}(\theta) \\ &= p + (a\mathbf{i} + b\mathbf{j}) \cdot (\cos\theta\mathbf{i} + \sin\theta\mathbf{j}) \\ &= p + a\cos\theta + b\sin\theta = p + t\cos(\theta - \alpha), \end{aligned} \tag{6a}$$

where

$$t = (a^2 + b^2)^{1/2}; \qquad \alpha = \tan^{-1}(b/a). \tag{6b}$$

The second half of Eq. 6a follows from a trigonometric identity. We hereafter describe translations by the parameters t and α. To evaluate and interpret (6), we consider two cases separately.

Case 1. $p + t\cos(\theta - \alpha) \geq 0$.

In this case, if the point **P** voted for a point (θ, p) in the Hough domain, the same vote would occur at $(\theta, p + t\cos(\theta - \alpha))$ in $H_t(\theta, p)$. Therefore, the elements of the column corresponding to θ in the original Hough array are shifted along the positive p axis by an amount equal to $t\cos(\theta - \alpha)$.

Case 2. $p + t\cos(\theta - \alpha) < 0$.

In this case the vote does not occur at $(\theta, p + t\cos(\theta - \alpha))$ since $p + t\cos(\theta - \alpha) < 0$. (Recall that in the Hough space, $p \geq 0$.) However, this implies that $-\mathbf{P}_t \cdot \mathbf{a}(\theta) = \mathbf{P}_t \cdot \mathbf{a}(\theta + \pi) = -(p + t\cos(\theta - \alpha)) > 0$ and therefore the vote would be entered at $(\theta + \pi, -p - t\cos(\theta - \alpha))$ in the new Hough space.

Combining these two cases, we can obtain $H(\theta, p)$ from $H_t(\theta, p)$ as

$$H(\theta, p) = \begin{cases} H_t(\theta, p + t\cos(\theta - \alpha)) & \text{if } p + t\cos(\theta - \alpha) \geq 0 \\ H_t(\theta + \pi, -p - t\cos(\theta - \alpha)) & \text{if } p + t\cos(\theta - \alpha) < 0. \end{cases} \tag{7}$$

These results show that a translation of the object causes shifts in the Hough transform in the vertical (p) direction only. The amount of the shift is a function of θ for each object point, i.e., it varies along the horizontal θ axis in the Hough space. For each column with a positive shift, there is a corresponding column a circular distance π away in θ that requires an equal negative shift. This occurs because $t\cos(\theta - \alpha + \pi) = -t\cos(\theta - \alpha)$. Thus half of the columns in $H_t(\theta, p)$ will have positive shifts and half of them will have corresponding negative shifts when we produce $H(\theta, p)$ from $H_t(\theta, p)$. Those elements that are shifted out of the Hough space as a result of the negative shifts reenter the Hough space a circular distance π away, we explained in Case 2.

3.5. *Combined Scale, Rotation, and Translation Transformation*

Equations (4), (5), and (7) can be combined to yield

$$H(\theta, p) = \begin{cases} H'\left(\theta + \phi, \dfrac{p + t\cos(\theta - \alpha)}{s}\right) & \text{if } p + t\cos(\theta - \alpha) \geq 0 \\ H'\left(\theta + \phi + \pi, \dfrac{-p - t\cos(\theta - \alpha)}{s}\right) & \text{if } p + t\cos(\theta - \alpha) < 0. \end{cases} \tag{8}$$

This relates $H'(\theta, p)$ for a general distortion to $H(\theta, p)$. In Eq. (8), it is understood that the additions to θ are performed modulo 2π.

3.6. *Digital Implementation of Distortion Transformations*

A digital implementation of the distortion transformations is particularly simple. Assume that $H'(\theta, p)$ is stored as a 2-D array and that the translation of the object is known. To undo the distortion in $H'(\theta, p)$ caused by translation, we need merely shift the columns corresponding to different θ by an amount $t\cos(\theta - \alpha)$. Since t and α are known, the amount of shift for each θ can be precomputed. If we feed each element in the top row of the new $H'(\theta, p)$ to the element in the same row a distance $\theta = \pi$ away horizontally, then as the elements of $H'(\theta, p)$ are shifted out from the top row in one column, they enter the proper column a distance $\theta = \pi$

away, causing downward shifts in these columns. This follows from the earlier discussion of Eq. (7). This is easily achieved by up/down shift register type memories.

Having corrected the effects of translation as above, the $H'(\theta, p)$ distortion effects due to rotation ϕ are similarly corrected by circularly shifting the rows of $H'(\theta, p)$ by ϕ in the θ direction.

To produce $H(\theta, p)$ from $H_s(\theta, p)$ for a scaled input and a given s, we consider two cases (depending on whether $s > 1$ or $s < 1$). We assume that p and s or $1/s$ are integers. (The implementation is a little more involved if s is not an integer and will not be discussed in this paper).

Case 1. $s > 1$ (compressed image).

Assume that s is an integer. $H_s(\theta, p/s)$ is defined only for those values of p for which p/s is an integer. Thus, using (4) we produce $H(\theta, p)$ from $H_s(\theta, p)$ for p such that p/s is an integer. The remaining rows in $H(\theta, p)$ are assigned zero values. Thus, we produce $H(\theta, p)$ from $H_s(\theta, p)$ by (4) for rows p where p/s is an integer and by inserting zero-valued rows in the appropriate rows p of the array where p/s is not an integer. This operation is also easily achieved in advanced memory arrays.

Case 2. $s < 1$ (expanded image).

Here we replace s by $1/s$ (an integer). From (4), for the case of a scale change, we require $H_s(\theta, sp) = H(\theta, p)$ and $H_s(\theta, s(p + 1)) = H(\theta, p + 1)$ for all p. Consider row r in $H_s(\theta, p)$ such that $sp < r < s(p + 1)$. Since r is not exactly divisible by s, no row in $H(\theta, p)$ exactly corresponds to this row in $H_s(\theta, p)$. Therefore, we add the votes for this row to the nearest discretized value of r/s (either p or $p + 1$). Thus, to obtain $H(\theta, p)$ from $H_s(\theta, p)$ for a given s, we need merely shift the data in all rows r in $H_s(\theta, p)$ (for which r/s is not an integer) and add these data to the data in the closest rows that are divisible by s. These scale distortion transformation can also be easily implemented using shift and add memory techniques.

4. HIERARCHICAL MATCHING

In the previous section, we described a method of efficiently producing the Hough transform of the image for a given scale, rotation, and translation. The method assumes that the scale, rotation and translation parameters are known. In practice, we are given a reference image and are required to estimate these parameters for an input image. In this section, we address simple techniques to estimate these parameters.

4.1. Brute Force Method

One method uses brute force. In this method, we consider all probable combinations of these distortion parameters and for each of these allowable combinations, we construct the associated Hough transform from the observed Hough transform of the input image. The combination of distortion parameters that give an $H(\theta, p)$ that best matches that of the reference(s) yields the distortion estimates and the object class estimate. If the number of possible combinations of distortion parameters is large, the brute force method will be slow and inefficient.

4.2. Reduced Distortion Parameter Search

We thus consider methods and cases when the number of possible distortion parameter combinations can be reduced. This is very application-specific of course. If the target range data (or a range image) is available, the value of scale can be estimated quite accurately. If the application concerns top–down views of objects such as aircraft, then the orientation and location in $H(\theta, p)$ of the two parallel lines that define the fuselage of the aircraft provide a good estimate of the object's rotation. Additional object distortion information is easily obtained from simple operations on the image. For example, the translational location of the object can be determined from the projections of the image along the x and y axes or from the first order moments m_{01} and m_{10}.

4.3. Hierarchical Search and Classification Method

We now detail a simple three-level hierarchical matching–search procedure that we have found to work well when the scale of the object is known and when the object is approximately centered (using moments or projections). Figure 5 shows this method in block diagram form. We describe this processor with the distortion transforms (Sect. 3) applied to the reference patterns. In the first level, the translation is ignored and the Hough transform of the input object is matched with all allowed rotated versions of the Hough transform of each reference object. This search is performed for rotations ϕ quantized in $\Delta\phi$ intervals to the degree desired and required for the given object classes and application. This can be easily achieved by feeding the Hough transforms of the input and reference images to a 1-D correlator as shown in Fig. 5. This is because a rotation of the object gives rise to a corresponding 1-D shift along the ϕ axis in the Hough domain (Sect. 3.3). The rotation angle ϕ_1 corresponding to the best match and its two nearest neighbors ϕ_2 and ϕ_3 are retained as the three most probable ϕ values. From the centering accuracy possible, the maximum value of t, $t_{\max}$, is known. In the second level, a value for t is assumed. (We use $t_{\max}/2$). We must still search the distortion transforms of the reference objects(s) for all expected α values for each of the three

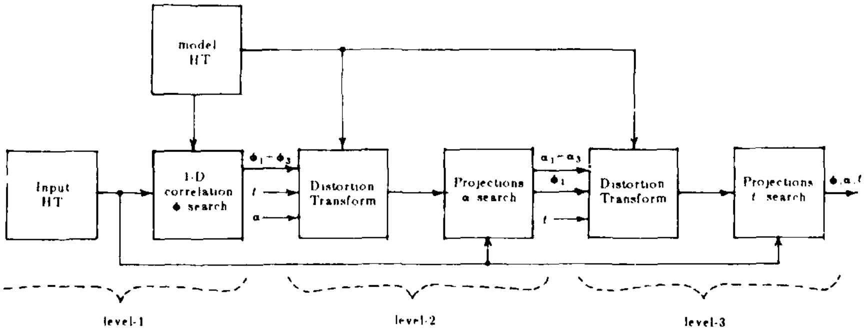


FIG. 5. Block diagram of the HT hierarchical search and classification method

ϕ estimates obtained from the previous level. This can be easily achieved by applying only the α distortion transforms to the Hough transforms of the reference objects (Sect. 3.6). A different α value results in a new HT that is not simply a 1-D shifted version of the original HT. Thus, this matching in the Hough space can be done by multiplying the corresponding elements of the Hough transforms and adding the products. This amounts to evaluating the correlation value at the center point. In Fig. 5, these correlations evaluated at one point are referred to as projections. The $\Delta\alpha$ quantization used is determined by the object classes involved and the accuracy required in the given application. The ϕ value and three α values corresponding to the best match (α_1) and its two nearest neighbors ([illegible] and α_3) are passed to level 3. In level 3, we search t from 0 to t_{max} for the three α values and the one best ϕ value determined from level 2. The HT for a new t value is again a new HT and this search in Δt increments is performed as the α search in level 2 was. The number of t values and the range of t to be searched are set by the expected accuracy of the centering method used. The best match yields the final t, α, ϕ, and object class estimates. This concept can be extended to include a scale search as well, with an associated increase in complexity. Section 5 details and quantifies this hierarchical procedure for different aircraft image classification problems with attention to the quantizations $\Delta\phi$ and $\Delta\alpha$ and the number of searches needed.

5. DATABASE AND INITIAL TEST RESULTS

5.1. Database

The images used in our initial experiments were top-down edge (boundary) images of five different types of aircraft with a resolution of 128 × 128 pixels. Figure 6 shows the $\phi = 0$ edge images of the five aircraft types used. Using specialized software and aircraft model descriptions, various translated versions of each image with t varied from 0 to 60 pixels within a 256 × 256 pixel image frame were used together with different rotated and scaled versions of each image with the scale $s = 1$, 2, and 3. For test inputs, t was varied continuously from 0 to 60, whereas our t quantization used in the system was 10 pixels. Thus our t estimates are expected to be accurate only to ± 5 pixels. The α translation parameters used ranged from 0 to 315° and were quantized to $\Delta\alpha = 45°$. The rotation parameters ϕ

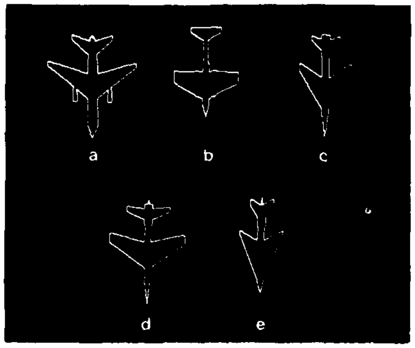


FIG. 6. Edge images of the aircraft types used. (a) DC10, (b) B57, (c) F105, (d) Mirage, (e) Mig

used varied from 0° to 330° in increments $\Delta\phi = 30°$. Thus, there are 12 values of ϕ, 3 values of s, 6 nonzero values of t, and 8 values of α for each nonzero value of t. This makes a total of $(1 + 6 \times 8)12 \times 3 = 1764$ possible combinations of the distortion parameters.

The $H(\theta, p)$ transform space was computed as in (1) with $\Delta\theta = 5°$, $\Delta p = 5$ and the origin in the center of the 256×256 image frame. The Hough array (θ, p) is thus of size $360/5 \times 128/5$ or 72×26. Byte arrays were used to store $H(\theta, p)$ to 256 levels from 0 to 255. The largest pixel value in all $H(\theta, p)$ arrays was normalized to 255 and values below a threshold were simply set to zero to reduce the computations in the matching process. A threshold of 40 was used for noise-free images. The hierarchical search test results involving scale changes have not been included in this paper. However, our experiments indicate that in order to achieve good results with scaled images, we need to compute the Hough transform with slightly better resolution, $\Delta\theta = 2°$ and $\Delta p = 2$.

5.2. *Representative $H(\theta, p)$ Examples*

In Fig. 7a we show $H(\theta, p)$ for a Mirage oriented at $\phi = 0$ and centered at the origin and in Fig. 7b we show $H(\theta, p)$ for the Mirage shifted upwards by 60 pixels. We discuss Figs. 7a and b to provide insight on the contents and pattern in the Hough transform. The peaks in each $H(\theta, p)$ can be associated with the various lines in the image. In Fig. 7a, the bright peaks at approximately $\theta = 270° \pm 30°$ correspond to the two lines that define the front edge of the wings, the peaks near $\theta = 90°$ correspond to the back edges of the wings and the edges of the tail. The two parallel vertical lines that define the fuselage produce peaks at $p = 0$ and $\theta = 0$ and 180°. (Recall that p was discretized to integer multiples of 5.) In the Hough transform in Fig. 7b of the Mirage translated vertically upward by 60 pixels, the columns of the Hough array are shifted up or down by $t\cos(\theta - \alpha) = 60\cos(\theta - 90°) = 60\sin\theta$, as in Eq. (6). The shifts from $\theta = 0$ to 180° are positive downward with the maximum shift occurring at $\theta = 90°$. The shifts for θ between π

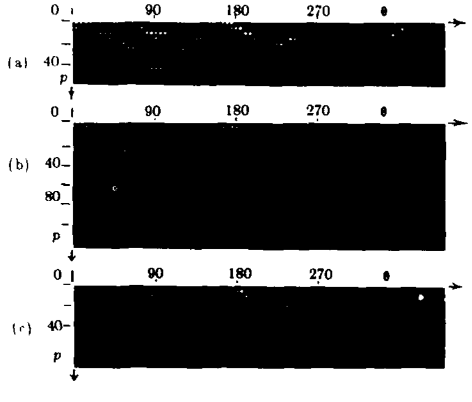


FIG. 7. The Hough transform of the Mirage (a) centered at the origin, (b) shifted upwards by 60 pixels, and (c) corresponding to the best match. For this test, $C_4 = 2.38 \times 10^6$, $C_1 = 1.51 \times 10^6$, and $C_2 = 1.04 \times 10^6$.

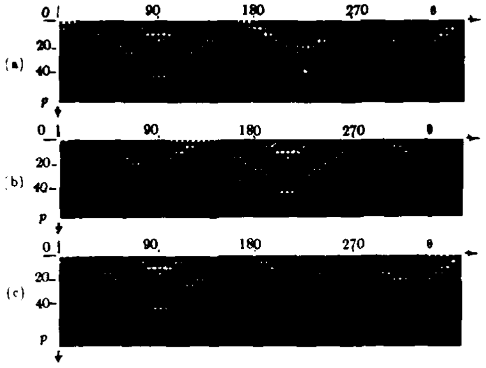


FIG. 8. The Hough transform of the centered Mirage (a) unrotated, (b) rotated by 120°, and (c) corresponding to the best match. For this test, $C_A = 2.38 \times 10^6$, $C_1 = 2.02 \times 10^6$ and $C_2 = 1.62 \times 10^6$.

and 2π are negative and the original data there merges in the top portion of the array and enters 180° away between $\theta = 0$ and π. As seen, this causes the peaks due to the front edges of the wings to now occur at 90° ± 30° (at smaller p values) instead of at smaller p values at 270° ± 30° as in Fig. 7a.

To determine the distortion of this one known class of input test object from Fig. 7b, we could produce $H(\theta, p)$ for all 1764 possible sets of the distortion parameters (s, t, α, and ϕ) applied to the Hough space. For each case, the new $H'(\theta, p)$ could be template matched against the $H(\theta, p)$ reference in Fig. 7a. The distortion parameters associated with the largest correlation value obtained are selected as the best estimate. Figure 7c shows $H'(\theta p)$ for the best match. As can be seen, it is visually very similar to the original $H(\theta, p)$ in Fig. 7a. The correlation value $C_1 = 1.51 \times 10^6$ for the correct $(t, \alpha, \phi) = (60, 90°, 0)$ choice was the largest one obtained. The next largest value $C_2 = 1.04 \times 10^6$ occurred for $(t, \alpha, \phi) = (50, 90°, 0)$. The maximum C_1 value compares favorably with the autocorrelation $C_A = 2.38 \times 10^6$ of the original $H(\theta, p)$. Thus, local maxima can be avoided and high confidence in the final estimate can be obtained by ensuring that C_1 is some high fraction of C_A (typically $\approx 0.6\ C_A$).

Figure 8 shows similar one-class test results for the Mirage with $\phi = 0°$ (Fig. 8a) and $\phi = 120°$ (Fig. 8b) rotation only. The $H'(\theta, p)$ pattern with the best match is shown in Fig. 8c with its C_1 value and the associated (t, α, ϕ) parameters. The C_2 value for the next best match is listed for completeness. Again, the correct object distortion estimates are obtained. The variations in the C_1 values arise due to the quantization of the Hough space. Visual inspection of Figs. 8a and b shows that they are the same with Fig. 8b being a cyclically shifted version of Fig. 8a (with a cyclic shift of 120° or 120°/360° = $\frac{1}{3}$ of the $H(\theta, p)$ pattern. As can be seen, Fig. 8c is almost identical to Fig. 8a.

5.3. Multiple-Distortion Intra-Class Recognition Tests

This $H(\theta, p)$ transformation and template matching technique was then applied to multi-class multiple-distorted versions of the five aircraft types. Columns 1–4 in

TABLE 1

Selected Intra-Class Multiple Distortion Test Results

				Best estimates from full search $\Delta\phi = 30°, \Delta t = 10, \Delta\alpha = 45°$		Best estimates from hierarchical search $\Delta\phi = 30°, \Delta t = 10, \Delta\alpha = 45°$	
Test no.	Aircraft name	Translation a, b (pixels)	Rotation ϕ (degrees)	Translation a, b (pixels)	Rotation ϕ (degrees)	Translation a, b (pixels)	Rotation ϕ (degrees)
1	Mirage	0, 60	0	0, 60	0	0, 60	0
2	Mirage	−30, 30	0	−28, 28	0	28, −28	30
3	Mirage	14, 14	120	14, 14	120	14, 14	120
4	DC10	7, 7	270	7, 7	270	7, 7	270
5	DC10	−25, −25	270	28, 28	270	14, 14	150*
6	DC10	30, 35	270	35, 35	270	35, 35	150*
7	B57	5, 5	320	7, 7	330	7, 7	330
8	B57	17, 17	320	21, 21	330	21, 21	330
9	B57	58, 5	320	60, 0	330	50, 0	30*
10	Mig	8, 8	225	7, 7	210	7, 7	210
11	Mig	14, 14	225	14, 14	210	14, 14	210
12	Mig	45, 41	225	42, 42	210	21, 21	270*
13	F105	9, 9	330	14, 14	330	14, 14	330
14	F105	20, −20	330	21, 21	330	14, 14	210*
15	F105	60, 5	330	60, 0	330	60, 0	300

*Large t ($t > 25$)

Table 1 describe the input test data. Data for three representative distorted versions of each aircraft type are included. These initial one class (intra-class) results assume that the object class was known and thus only represent tests of distortion parameter (s, ϕ, a, b) estimates. The results for both a full (brute force) search and our hierarchical search are included. The full search method results (columns 5 and 6 of Table 1) always yield the correct estimates within the quantizations $\Delta\phi = 30°$, $\Delta t = 10$, and $\Delta\alpha = 45°$ of our distortion parameters. The estimates for translation are given in terms of the a and b parameters which can be easily obtained from the t and α parameters.

The results using our hierarchical search method are now discussed. Note that the test inputs are only approximately centered in these tests. The intra-class test results on the same 15 test images using our hierarchical search method are presented in columns 7 and 9 of Table 1. The scale s is assumed to be known. In the first level, 12 tests of ϕ are made ($\Delta\phi = 30°$) assuming that the translation is zero (i.e., $a = b = 0$) and the three best values are passed to the second level. In the second level, 8 values of α are tested for the three best ϕ values from level 1 (i.e., $8 \times 3 = 24$ tests are performed). These level 2 tests are performed for a fixed $t = (a^2 + b^2)^{1/2} = 20$. Since the object is assumed to be approximately centered, $t = 20$ is a reasonable estimate for translation. The three best α values and the best ϕ value are then passed to level 3, where six t tests for each α are made ($3 \times 6 = 18$ tests). The total number of test matchings required is thus $12 + 24 + 18$ or only 54. This is a significant reduction from the 1764 tests required in the brute force method. As can be seen from the results, this method gave comparable results, except for large t ($t > 25$) denoted by * in Table 1. This is expected because the simple method (assuming $t = 0$) used to estimate the ϕ value in the first level failed. By centering the object in advance or by including several t values in level 1, near perfect performance can be obtained.

5.4. Discrimination and Multiple-Distortion Performance

Table 2 shows test results of the discrimination and recognition performance of our hierarchical method in a multi-class case. Columns 2–4 list the selected input test image information. The tests included four of the input aircraft types with different multiple translation (a, b) and rotation (ϕ) distortions present and one (test 5) with only a shift. The best template match for each test input with two to four of the reference aircraft types is given (columns 5–8). In tests 1–3, we see that both the correct aircraft class and the correct distortion parameters are obtained. Such an excellent performance is expected when $t < 25$. Thus the multi-class discrimination and intra-class recognition (multiple distortion invariance) features of this processor have been demonstrated. From Fig. 6, we see that the F105 and Mig images are rather similar. We thus expect discrimination between these two aircraft types to be difficult. In test 4, we find that the Mig input would be misclassified as an F105. Using a Hough array with higher spatial resolution could resolve these two similar classes. If we use the fact that $C_4 = 0.83 \times 10^6$ occurs for the Mig and that a larger $C_4 = 1.56 \times 10^6$ occurs for the F105, we can normalize the data or set $C_1 = 0.6 \times C_4$ and realize that the observed C_1 is too large and thus

TABLE 2

Multi-Class Multiple Distortion Recognition and Performance of Hierarchical Hough Transform Transformations and Matching

Input test aircraft information				Hierarchical processor results Best estimates $\Delta\phi = 30°$, $\Delta t = 10$, $\Delta\alpha = 45°$			
Test no.	Aircraft type	Translation a, b (pixels)	Rotation ϕ (degrees)	Reference aircraft	a, b	ϕ	Correlation value
1	DC10	7, 7	270	DC10	7, 7	270	1.77×10^6
				F105	7, 7	270	1.27×10^6
				B57	20, 0	240	1.03×10^6
				Mirage	0, 0	270	1.61×10^6
2	B57	7, 7	30	B57	−7, −7	30	1.53×10^6
				DC10	−14, −14	0	1.14×10^6
				F105	14, 14,	60	1.00×10^6
3	F105	−7, 7	330	F105	−7, 7	330	1.45×10^6
				DC10	0, 10	330	1.30×10^6
				B57	7, 7	330	1.11×10^6
				Mig	−10, 0	330	1.00×10^6
4	Mig	−14, −14	150	F105	−21, −21	150*	1.05×10^6
				Mig	−14, −14	150	0.79×10^6
5	Mirage	0, 60	0	Mirage	0, 60	0	1.51×10^6
				DC10	0, 60	0	1.16×10^6

* Misclassified.

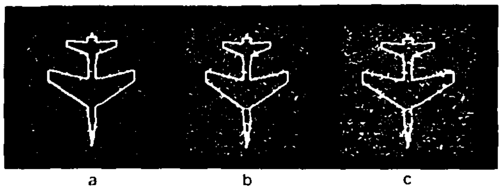


FIG. 9. Image of the Mirage with noise with (a) $\sigma_n = 0.2$, (b) $\sigma_n = 0.25$, and (c) $\sigma_n = 0.3$

provide discrimination of such similar object classes. From Fig. 6, we also note that the Mirage and DC10 are similar in shape as well as in size. Test 5 was included to show that our Hough transform hierarchical technique still allows us to discriminate between them. All the test results were the same when the brute force method was used, i.e., identical values for the best C_1 and C_2 values were obtained.

5.5. *Noise Performance*

To determine and quantify the performance of these methods in the presence of noise, noisy input images were generated as follows. Random noise with a Gaussian distribution and of zero mean and different variance σ_n was added to each pixel in the test image. The resulting image was then rebinarized by thresholding it at 0.5. Figure 9 shows the image of the Mirage when noise with $\sigma_n = 0.2$, $\sigma_n = 0.25$, and $\sigma_n = 0.3$ was added. Table 3 shows the performance of our full search and hierarchical methods for intra-class multiple distortion estimation with a noise variance $\sigma_n = 0.2$. As seen, all results are perfect in the case of the brute force method (within our quantization). The results in the case of the hierarchical method are correct except in the case of test 3. When σ_n was increased to 0.3, the brute force method still gave the same results, but the hierarchical method was in error in 30–50% of the cases with the ϕ estimate in level 1 generally being the estimation parameter in error.

TABLE 3

Selected Intra-Class Multiple Distortion Test Results When Noise with $\sigma_n = 0.2$ Was Added

				Best estimates from full search $\Delta\phi = 30°$, $\Delta r = 10$, $\Delta a = 45$		Best estimates from hierarchical search $\Delta\phi = 30°$, $\Delta r = 10$, $\Delta a = 45$	
Test no	Aircraft name	Translation a, b (pixels)	Rotation ϕ (degrees)	Translation a, b (pixels)	Rotation ϕ (degrees)	Translation a, b (pixels)	Rotation ϕ (degrees)
1	Mirage	0, 60	0	0, 60	0	0, 60	0
2	Mirage	14, 14	120	14, 14	120	20, 0	30*
3	DC10	7, 7	270	7, 7	270	7, 7	270
4	B57	5, 5	320	7, 7	330	7, 7	330
5	B57	17, 17	320	21, 21	330	21, 21	330
6	Mig	8, 8	225	7, 7	210	7, 7	210
7	Mig	14, 14	225	14, 14	210	14, 14	210
8	F105	9, 9	330	14, 14	330	14, 14	330
9	F105	60, 5	330	60, 0	330	50, 0	300

* Wrong parameter estimates

TABLE 4

Multi-class Multiple Distortion Recognition and Performance of Hierarchical Hough Transform Transformations and Matching When Noise with $\sigma_n = 0.25$ Was Added

Input test aircraft information				Hierarchical processor results			
				Best estimates $\Delta\phi = 30°, \Delta r = 10, \Delta\alpha = 45°$			
Test no	Aircraft type	Translation a, b (pixels)	Rotation ϕ (degrees)	Reference aircraft	a, b	ϕ	Correlation value
1	DC10	7, 7	270	DC10	7, 7	270	1.51×10^6
				F105	7, 7	270	1.09×10^6
				B57	0, 0	270	1.06×10^6
				Mirage	0, 0	270	1.43×10^6
2	B57	7, −7	30	B57	7,	30	1.34×10^6
				DC10	0, 0	30	1.29×10^6
				F105	0, 0	120	1.06×10^6
3	F105	7, 7	330	F105	7, 7	330	1.51×10^6
				DC10	0, 10	330*	1.54×10^6
				B57	7, 7	330	1.26×10^6
				Mig	10, 0	330	1.03×10^6
4	Mig	14, −14	150	F105	21, −21	150	1.39×10^6
				Mig	14, −14	150	0.99×10^6
5	Mirage	0, 60	0	Mirage	0, 60	0	1.31×10^6
				DC10	0, 60	0	1.09×10^6

* Misclassified

When discrimination performance with multiple distortions for $\sigma_n = 0.2$ was tested the results obtained were essentially the same as those for the noise-free cases in Table 2. However, when σ_n was increased to 0.25 (Table 4), the method is found to make an additional error, with the F105 being wrongly classified as a DC10 (test 3, Table 4). A threshold of 80 in the Hough space was used for these noise tests.

The signal to noise ratio (SNR) in these tests can be computed as

$$\mathrm{SNR} = \frac{N_b}{N_{n1} + N_{n2}}, \tag{9}$$

where N_b is the number of boundary pixels on the noise-free target, N_{n1} is the number of background pixels added and N_{n2} is the number of target pixels removed. For $\sigma_n = 0.2(0.25)$ and the Mirage aircraft, SNR = 1.17(0.316). Thus, our observed performance is excellent in the case of poor input SNR.

6 SUMMARY AND CONCLUSIONS

A new approach using the basic Hough transform defined for straight lines has been suggested for estimating the scale, translation and rotation distortion parameters of an input test object. The method is capable of multi-class object discrimination and multiple-distortion object recognition. Test results on aircraft imagery were provided and shown to be excellent for multi-class discrimination, distortion parameter estimation and in the presence of noise. The new direct use of the Hough space is possible by use of the new and efficient Hough transform distortion transformations developed. A new hierarchical search method was devised that allows efficient realization of the proposed concept. This technique also allows the Hough space to be spatially quantized, thereby further simplifying realization. If the translation of the object is large, the use of moments (or similar methods) to center the object, combined with a 1-D correlation and followed by matching with a few distortion-transformed images provides the class, scale, rotation and translation estimates. For the accurate estimation of scale, a higher spatial resolution in the Hough space is required.

ACKNOWLEDGMENTS

The support of this research by DARPA and AFOSR is gratefully acknowledged. The use of the facilities of the Center for Excellence in Optical Data Processing at C-MU (supported by a Westinghouse Corporation Equipment Grant) is also acknowledged.

REFERENCES

1. P. V. C. Hough, *Method and Means for Recognizing Complex Patterns*, U. S. Patent 3 069 654, 1962.
2. R. O. Duda and P. E. Hart, Use of the Hough transform to detect lines and curves in pictures, *Comm. ACM* **15**, 1972, 11–15.
3. D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* **13** (2), 111–122.
4. D. H. Ballard and D. Sabbah, Viewer independent shape recognition, *IEEE Trans. Pattern Anal. Mach. Intelligence* **5** (6), 1983, 653–660.
5. I. M. Silberberg, L. Davis, and D. Harwood, An iterative Hough procedure for three dimensional object recognition, *Pattern Recognition* **17** (6), 1984, 277–285.
6. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice Hall, Englewood, N.J., 1982, Chap. 4.
7. L. S. Davis, Hierarchical generalized Hough transforms and line-segment based generalized Hough transforms, *Pattern Recognition* **15** (4), 1982, 277–285.
8. C. M. Brown, Inherent bias and noise in the Hough transform, *IEEE Trans. Pattern Anal. Mach. Intelligence* **5** (5), 1983, 493–505.
9. S. R. Deans, Hough transform from the Radon transform, *IEEE Trans. Pattern Anal. Mach. Intelligence* **3** (2), 1981, 185–188.
10. G. Eichman and B. Z. Dong, Coherent optical production of the Radon transform, *Appl. Optics* **22** (6), 1983, 830–834.
11. G. R. Gindi and A. F. Gmitro, Optical feature extraction via the Hough transform, *Optical Engrg.* **23** (5), 1984, 499–506.
12. P. M. Merlin and D. J. Farber, A parallel mechanism for detecting curves in pictures, *IEEE Transa. Comput.* **24** (1), 1975, 96–98.
13. H. Li, M. A. Levin, and R. J. LeMaster, *Fast Hough Transform*, Research Report No. RC 11080 (#49754), Manufacturing Research, IBM, NY, 29 March 1985.

3. LARGE CLASS ICONIC OPTICAL PROCESSING

Reprinted from Applied Optics, Vol. 26, Page 2266, June 1, 1987
Copyright © 1987 by the Optical Society of America and reprinted by permission of the copyright owner.

Optical iconic filters for large class recognition

David Casasent and Abhijit Mahalamobis

Approaches are advanced for pattern recognition when a large number of classes must be identified Multilevel encoded multiple-iconic filters are considered for this problem. Hierarchical arrangements o iconic filters and/or preprocessing stages are described. A theoretical basis for the sidelobe level and noise effects of filters designed for large class problems is advanced. Experimental data are provided for an optica character recognition case study.

I. Introduction

Advanced artificial intelligence, symbolic, and other processors required to operate on large knowledge bases[1,2] need techniques to handle a large number of object classes. We consider pattern recognition applications when the number of object classes to be identified is large. Our approach can be applied to logic processors (in which the input is a query) and to symbolic and associative[3] processors. However, pattern recognition offers a more easily defined problem, and thus we pursue this specific application. We employ an optical character recognition (OCR) case study example to quantify and demonstrate remarks and results, since such a data base is easily available. Much recent pattern recognition research has addressed algorithms to achieve distortion-invariance, i.e., recognition of geometrically distorted versions of an object.[4-6] In this paper we consider large class problems in which the number of different objects is large. Incorporation of distortion-invariant techniques into the filters we discuss can further broaden their use. Since the filters we discuss operate on input image pixel representations, we refer to them as iconic filters.[7]

Section II describes our OCR data base, and Sec. III reviews several basic iconic filter synthesis algorithms. In Sec. IV we advance a theoretical analysis of the effect of the number of training images and object classes on the output sidelobe level and the noise sensitivity of iconic filters. Section V describes several systems to achieve large class recognition without the iconic filter problems associated with large training sets of data. Experimental data are then provided (Sec. VI) to quantify and demonstrate all major points advanced.

II. Data Base and Case Study

As an easily obtainable data base we selected recognition of the 62 characters (26 lower-case and 26 upper-case letters, plus the 10 number digits) in a variety of fonts. We obtain 80 × 80 pixel images of the 62 characters from 15 different magazines: *Time*, *Scientific American* (Scienam), *Datamation* (Datama), *Business Week* (Busweek), etc. We will refer to the fifteen versions of each character as fonts (although they represent different point sizes of each character as well) In our experiments, we will view these as in-class variations. Font identification can be achieved by other methods.[8] Our filters are thus designed to be able to provide the recognition of each character independen of the input font, but without the requirement to identify the input data font. This choice also allows us tes data that are not present in the training set used to synthesize the filters. Figure 1 shows several characters from three of the magazines to demonstrate th similarity and differences in the fonts present in ou data base.

III. Iconic Filter Synthesis

The basic filters considered are extensions of on type[9,10] of distortion-invariant matched spatial filter with attention to our present application. For com pleteness we review three types of these filters an three classes of filters possible. This section also a lows the terminology to be defined.

We denote objects in one class by $\{f_n\}$ and objects in second class by $\{g_n\}$. The members within each class are generally different 3-D geometrically distorte versions (e.g., aspect views) of each object. In ou

---

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Pittsburgh, Pennsylvania 15213.

Received 10 October 1986.
0003-6935/87/112266-08$02.00/0.
© 1987 Optical Society of America.

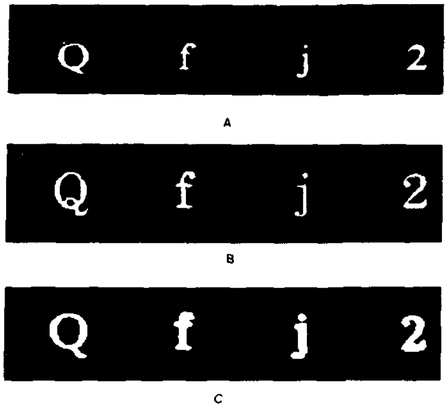


Fig. 1. Typical characters from three different publications: (a) *The New York Times*; (b) *Datamation*; and (c) *Scientific American*.

OCR application the members within each class will be different font representations of each input character/object. We denote vector versions (e.g., lexicographically ordered images) of the objects by $\mathbf{f}_n$ and $\mathbf{g}_n$ and the filters designed by $\mathbf{h}_k$ (all are 2-D images, or vectors). When $\mathbf{f}_n$ and $\mathbf{g}_n$ are similar (such a filter to recognize one class must also have information on the other class), we specify a filter $\mathbf{h}$ so that

$$\langle \mathbf{f}_n \cdot \mathbf{h} \rangle = 1, \langle \mathbf{g}_n \cdot \mathbf{h} \rangle = 0 \tag{1}$$

for all n, where ⟨ ⟩ denotes the vector inner product operation $\mathbf{f}^T\mathbf{h}$. We restrict all filters to be linear combinations of all training set images

$$h(x,y) = \sum_{n=1}^{N_1} a_n f_n(x,y) + \sum_{n=N_1+1}^{(N_1+N_2)} a_n g_n(x,y). \tag{2}$$

For N_1 images in $\{f_n\}$ and N_2 images in $\{g_n\}$, the $N_2 + N_2$ coefficients a_n define the filter function. The coefficient vector $\mathbf{a}$ and hence the filter function $\mathbf{h}$ are the solution of $\mathbf{V}\,\mathbf{a} = \mathbf{u}$, where $\mathbf{V}$ is the vector inner product matrix of the data set, and $\mathbf{u} = \mathbf{u}_1 = [1\ldots1, 0\ldots]^T$ is set by Eq. (1) to yield 1 outputs for all N_1 images in class one and 0 outputs for all N_2 images in class two. The filter is thus specified by

$$\mathbf{a} = \mathbf{V}^{-1}\mathbf{u}_1. \tag{3}$$

To recognize $\{\mathbf{g}_n\}$ and reject $\{\mathbf{f}_n\}$, the control vector $\mathbf{u}_1$ in Eq. (3) is simply changed to $[0\ldots0, 1\ldots1]^T$, and a new $\mathbf{a}$ set of weights is determined.

A multilevel filter with outputs equal to one for class one objects and two for class two objects can easily be fabricated using the control vector $\mathbf{u}_1 = [1\ldots1, 2\ldots2, 3\ldots3]^T$ in Eq. (3). As shown, extensions of this filter to more than two classes are possible. Binary-encoded multiple filters can also be employed. In this case the outputs from the filters define a digital word (e.g., 10, 01, 11, for the case of $F = 2$ filters) that denotes the object class (e.g., if the outputs from the two filters are both 1, the code word is 11 and the input test object is in class three). Synthesis of these filters uses the same basic technique in Eq. (3) with different $\mathbf{u}$ control vectors.

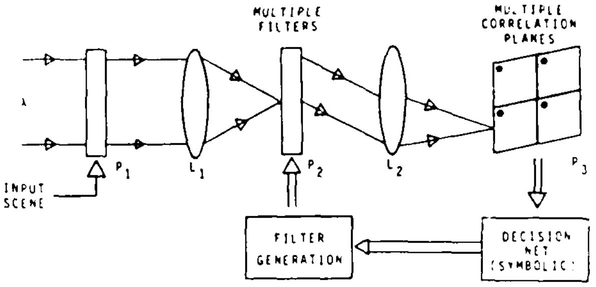


Fig. 2. Multichannel frequency plane correlator with $F = 4$ iconic matched spatial filters for large class pattern recognition.[11]

For large class problems we propose the use of multilevel multiple iconic filters (specifically F filters with L output levels). The output from such a system is now an F-digit word (one output/filter) and is thus capable of representing L^F different states or object classes (in practice $L^F - 1$ states are obtained since the all-zero state can also occur for no input object). Prior work on such filters has shown quite promising results. However, attention has been given to their distortion-invariance and no more than four object classes have been considered for use in such filters.

Three different classes of such iconic filters can be identified.[10] The filters described above are projection filters since the formulation specifies only the central or peak value in the correlation of $\mathbf{h}$ and the input object. For many object classes (especially when the total number of training images N_T is small), control of the central peak value in the correlation function allows sufficient performance and specially low sidelobe levels. We address this issue in detail in Sec. IV. For cases when the sidelobes for one object class are larger than the peak values for other classes (or larger than the value at the center of the correlation function for the same object class), correlation filters can be used. These filters[10] use shifted versions (typically four) of each training set image to control the shape of true correlation peaks (i.e., they specify a fixed value at the center of the correlation function and zero values at $\pm d_x$ pixels away, horizontally and vertically). These filters require five times the number of training images that are needed in the projection filter, and hence N_T effects for these filters will be worse. The best peak to sidelobe ratio (PSR) in the output correlation pattern is obtained with a PSR iconic filter.[10] The disadvantage of this filter is that its peak value cannot be specified. Thus since multilevel encoding is not possible with such a filter, the number of classes that one can accommodate using multiple PSR filters is significantly reduced.

These three filters are typically used as the filters in a frequency plane correlator. Figure 2 shows the classic frequency plane correlator with four frequency-multiplexed filters at P_2 and four output correlation planes at P_3. These $F = 4$ correlation planes are read out in parallel in raster format in synchronization. From the $F = 4$ digit output word obtained for each pixel location in the output correlation planes, the class or category of each region of the input image at P_1 can be obtained.[11] The use of more than four parallel correlation planes is generally prohibitive, and thus such an architecture can accommodate $L^F = L^4$ object classes. To accommodate large class problems, multilevel filters ($L > 2$) are thus essential.

These filters can also be applied to associative memories as detailed elsewhere.[12] The classic system is shown in Fig. 3. Here the input 1-D vector data **x** at P_1 describes an input object, and the F filters at P_2 are the columns of the associative memory matrix **M**. The P_3 output vector **v** is the F-digit encoding of the input object from which one can decode the object into a member of one of L^F classes.

IV. Large Training Class Effects on Iconic Filter Performance (Theory)

In numerous tests of the iconic filters described in Sec. III we noted that the performance of the projection and correlation filters degraded (i.e., large sidelobe levels occurred) as the number of training set images N_T was increased. For our large class problems of present concern N_T will also be large, and thus this issue is of significant concern. Thus we now address this issue theoretically for the case of correlation iconic filters. Solution of large matrices that arise in large class problems can be addressed by advanced techniques and is not of immediate concern here. The analysis is simplified by considering the Fourier transform of the correlation plane. Specifically, we consider the average (or mean) μ and scatter S of the magnitude of the Fourier transform of the correlation function. The average μ value equals the peak value in the correlation plane (this follows from Parseval's theorem)

$$\Sigma f(i)h(i) = (1/M)\Sigma F(k)H^*(k), \tag{4}$$

where f and h are 1-D sequences, and F and H are their Fourier transforms, and the summation is over the number of pixels M in each image. We thus write the average for an input image f_k and a linear combination filter h (described by coefficients a_n) as

$$\mu = E[H^*F_k] = \sum_n E[a_n F_n^* F_k] = \sum_n a_n v_{kn} = u_k, \tag{5}$$

where $v_{k,n}$ denotes element (k,n) of the matrix **V**, and u_k is element k of the control vector **u** in Eq. (3). The scatter S in the Fourier transform of the correlaton is a measure of the ripple or sidelobes present in the output correlation plane. Using Eq. (4) and the filter synthesis of Eq. (3), the scatter is shown to satisfy

$$S = E[|H^*F_k|^2] - \mu^2 \geq \sum_n \sum_m a_n a_m v_{nm} v_{kk} - \mu^2$$

$$S \geq v_{kk}(\mathbf{a}^T\mathbf{V}\,\mathbf{a}) - \mu^2. \tag{6}$$

We now consider how S varies as the number of training images N_T increases. Since the matrix **V** is symmetric and positive definite, we decompose it and easily show

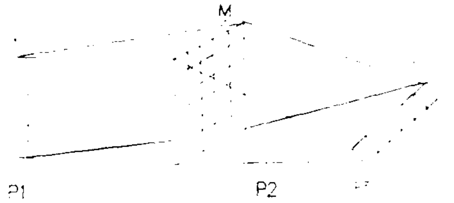


Fig. 3. Multiple iconic projection filter associative processor system.

$$\mathbf{a}^T\mathbf{V}\,\mathbf{a} = \sum_n \alpha_n^2\lambda_n, \tag{7}$$

where λ_n are the eigenvalues of **V**, and α_n^2 are positive constants. The term v_{kk} in Eq. (6) is positive (since these diagonal elements correspond to the autocorrelation of positive images). Similarly $\lambda_n \geq 0$ in Eq. (7) since **V** is a positive definite matrix. Although the terms $\alpha_n^2\lambda_n$ in Eq. (7) are positive, the values of the individual α_n and λ_n change with N_T. Hence for increasing N_T the sum in Eq. (7) [and hence the scatter in Eq. (6)] may increase or decrease. It can be shown that

$$\sum_n \alpha_n^2\lambda_n^2 = cN_T,$$

where c is a positive constant. This sum clearly increases with N_T and is an upper bound on Eq. (7). Thus the scatter S in Eq. (6) (and hence the correlation plane sidelobes) increases as the number of training images increases. Extensions of this theoretical treatment to the various other classes of iconic filters yield the same trend for the correlation sidelobes and the scatter S to increase with N_T.

In numerous tests we also observed (when more training images were used) that the dynamic range requirements of the filter and its noise requirements became more severe. We now advance a theoretical basis for this effect. We consider the average μ_F and the scatter S_F of the pixels in the filter image (denoted by the subscript F). The average and scatter now considered apply to the image plane representation of the filter function and not the output correlation plane. As S_F increases, the variations in the pixel values in the filter image itself increase and hence so does the number of levels required in the filter image and also the effects of noise (we will demonstrate this experimentally in Sec. VI). The mean of the filter image is

$$\mu_F = E[\mathbf{h}] = E\left[\sum_n a_n f_n\right] = \sum_n a_n E[\mathbf{f}_n] = \sum_n a_n v_{nn}/M, \tag{8}$$

where a linear combination filter is again assumed, and where the last equality in Eq. (8) is obtained by estimating $E[\mathbf{f}_n]$ by v_{nn}/M, where M is the number of pixels in the image. This approximation is realistic for our binary images, where v_{nn} is the dot product of image $\mathbf{f}_n$ and itself. From Eq. (8), the mean of the filter is thus seen to be proportional to the sum of the diagonal **V** weighted by the a_n linear combination filter coefficients.

Proceeding similarly, the scatter is found to be

$$S_F = E[\mathbf{h}^2] - E^2[\mathbf{h}] \tag{9a}$$

$$= (1/M)\left[\sum_n a_n^2 v_{nm}(1 - v_{nn}/M)\right.$$

$$\left. + 2\sum_n \sum_n a_n a_m (v_{nm} - v_{nn}v_{mm}/M)\right]$$

$$\simeq (1/M)\mathbf{a}^T\mathbf{V}\,\mathbf{a}. \tag{9c}$$

For cross products v_{nm} we have used a similar estimation for the expected value $E[\mathbf{f}_n\mathbf{f}_m] = v_{nm}/M$. The second double summation in Eq. (9b) does not include $n = m$. The final relation in (9c) assumes $1 - v_{nn}/M \simeq 1$ and $(v_{nm} - v_{nn}v_{mm}/M) \simeq v_{nm}$. These approximations are valid for our OCR character example, where the average auto projection value is $v_{nn} = 100$, and the average cross projection value is $v_{nm} \simeq 50$, and the number of pixels per image is $M = 6400$. From Eq. (7) we see that S_F in Eq. (9) increases with the number of training images N_T. This increases the filter's dynamic range. As we quantify in Sec. VI, this makes the effect of noise more significant in filters synthesized from a large number of training images N_T. In Sec. V we advance various ways to reduce N_T and yet achieve large class recognition.

V. Large Class Solutions

In this section we advance several solutions to the large class recognition problem with attention to the degraded performance of iconic correlation filters expected when a large set of training set images is used. In Sec. VI we advance experimental verifications of many of the suggested solutions. We note that our theory in Sec. IV applies not only to correlation filters, but also to projection filters if one does not look only at the correlation peak point. If projection filters are interrogated at the peak point only, the only limitation on N_T is in solving the synthesis Eq. (3). We will use this fact in several of our suggested solutions. Figure 4 shows the block diagram of a hierarchical iconic filter system.[11] The first stage of this processor employs multiple PSR filters in a shift-invariant correlator. The purpose of this first stage is only to locate candidate objects in the input field of view. The filters used are designed with this in mind, and thus they do not provide discrimination information. To provide enhanced detectability, PSR iconic filters are preferable for this stage of the processor. The second stage of the processor can employ multiple correlation or projection filters in the same processor. These filters allow large class identification (when multilevel outputs are provided), but they can have large sidelobe levels. By using the outputs from the PSR correlator in the first stage to determine where to look in the output correlation planes from the second stage, sidelobe effects can be avoided. In Fig. 4, we show a projection filter second stage, since it allows L^F class identification with F filters and with a simpler processor such as that of Fig. 3. This filter (and its associated matrix) also requires fewer training set images (less by a factor of 5) than are needed in the correlation iconic filter synthesis. An additional stage with correlation filters is often preferable in such a system, since some false peaks will occur in the first-stage processor, and the investigation of these points using only projection filters will force some object class decision for all regions of interest in the input scene (detected by the first filter stage). Error correlation[13] is another solution that can allow projection filters to be used directly without an additional stage of correlation filters to remove false region of interest peaks from the PSR filter.

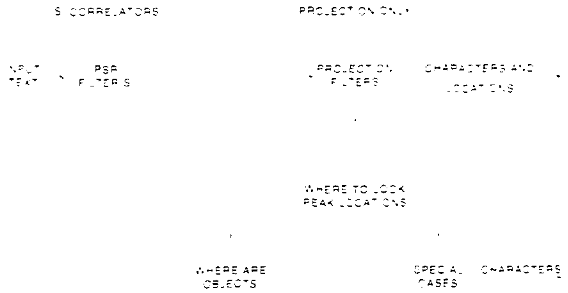


Fig. 4. Block diagram of a hierarchical iconic filter system for large class pattern recognition.

Another modification to the system of Fig. 4 is to perform feature space analysis in windows around the candidate region of interest areas indicated by the PSR iconic filters in the first stage. When F feature space discrimination functions are used and encoded in an F output L-level manner, a larger number of classes (L^F) can again be identified and classified. If we restrict analysis to only the central value of the output from the projection filters, these filters are in essence feature space linear discriminant functions that can operate on image pixel data (iconic filters) or on image features with equal facility.

In cases when the object size is known or can be bounded, the window around each region of interest image area can be set and simple techniques can be used to place the object in each region of interest into one of several super classes (e.g., one of 4 sets of 16 characters each). For the OCR case we have found simple object histograms and the number of pixels in the character and in different parts of it to work quite well to provide such super-class separation. Such information then allows the use of separate filters, each optimized on the smaller super class of possible objects and each with significantly fewer N_T training images. We have demonstrated iconic multilevel multiple filters in which the object class is known and the purpose of the filters is to determine the object orientation.[12] This represents yet another extension of this hierarchical filtering concept.

For a specific problem (such as OCR) other informatin is available such as: letters lie on lines with regular

Table I. Correlation Plane PSR = μ/S for Multilevel Multiple Iconic Correlation Filters as a Function of the Number of Object Classes.

Number of training images N_T (5/class)	Correlation plane PSR
10	2.04
20	1.98
40	1.76
60	1.48
100	1.52
200	0.98
400	0.01
930	0.006

Table II. Filter Image Plane Scatter S_F and Largest Pixel Value as a Function of the Number of Object Classes N_T for Different Multilevel Multiple Iconic Projection Filters.

N_T	S_F (scatter)	Maximum pixel value
2	0.02	0.05
15	0.03	0.06
25	0.18	0.10
35	0.35	0.22
75	0.78	0.82
115	0.87	0.95
130	0.89	0.96
150	0.92	1.29
170	1.05	1.62
190	1.16	1.66
248	1.33	2.31
930	18.10	9.90

spacings dependent on the font of the input data. For this case we find that simple horizontal and vertical projections can locate lines of text and isolate the letters on each line. In this case the center of each character can be determined quite simply with such a simple preprocessing step.

A related issue of concern is training set selection. In many cases attention to this issue can significantly reduce N_T. As an example we refer to our OCR case study with 15 fonts of each character available. We must select at least one image of each character. However, not all 15 fonts/character are required to be included in the training set. To select the fonts to be included we look at the cross correlations of each and select those with the smallest vector inner product matrix entry v_{mn}. This ensures us of the most new information for each additional training set image chosen. If the separation between output levels in a multilevel filter is ΔL, we select $v_{mn} \leq 0.5\Delta L$ as a useful guideline to determine when to include a given font image in our training set. In Sec. VI we show quantitative data on the ability of iconic filtrs to recognize characters in new fonts not included in the training set data.

VI. Experimental Results

To obtain a quantitative estimate of a number of object classes one can include in a correlation filter, multilevel multiple iconic correlation filters were computed with one object (character)/class or font and five shifted versions of each (thus $N_T/5$ equals the number of classes and fonts). For each case, μ and S of the FT of the correlation plane were calculated. The resultant PSR = μ/S is listed in Table I. Assuming PSR $\geq$ 1.5 is required, we find that only N_T = 100, or 20 object classes, could be included in one OCR correlation filter. We note that we have found that this value is much less for characters than for other objects, and thus OCR appears to represent a worst-case guideline.

To quantify the effect of N_T on the dynamic range of the filter and its image plane variance, we computed the mean μ_F and scatter S_F in the filter's image for multilevel multiple projection iconic filters with different numbers of training images used (with one image/class and with N_T now equal to the total number of object classes or fonts). These data are shown in Table II. In Table II we also include the value of the largest pixel in the iconic image plane filter. We note that the scatter (variance of the pixel values in the filter) increases with N_T. The maximum pixel value in the filter image increases with N_T. The number of filter image pixels with large values also increases with N_T. Thus more dynamic range or gray levels are required to represent filters synthesized with large N_T. Also, when noise is present, if the noise changes one of the large-valued (or key) image pixels, this will have a much larger effect than if other image pixels are changed. Since the number of such key pixels and their relative significance increases with N_T, we expect noise effects to become worse for large class filters synthesized from a large number of images. We now quantify this result and the amount of noise allowable.

The filters considered in subsequent tests were synthesized from 62 characters with 4 fonts of each (the fonts used were *NY Times*, *Datama*, *Busweek*, and *Forbes*). The multilevel multiple projection filters used F = 4 filters with L = 3 levels (0.33, 0.66, and 0.99), thus allowing $L^F = 3^4 = 81$ classes, which is sufficient to accommodate the 62 character classes. When these F = 4 filters were shown any of the 62 × 4 = 248 characters, the projection values were ideal and perfect 100% recognition was obtained. Table III shows the worst-case outputs (all are within 10^{-3} of the exact projection values).

We now consider the effect of noise on the performance of these filters. To produce the noise we generated a random array of numbers between 0 and 1. By thresholding this array at α, we produced a binary noise array $N(x,y)$ with pixels equal to 1 if their value was $\leq \alpha$. We then applied the same $N(x,y)$ to each character image with image pixels changed (0 to 1 or 1 to 0) if the corresponding (x,y) pixel in $N(x,y)$ is 1. We refer to the result as an image with binary noise. Test results for $\alpha = 0.5$ corresponding to $\sigma^2_{\text{noise}} = 0.25$ for the font *Busweek* are shown in Table IV. Only the worst-case results are shown (those data with projection values which departed by the most from the ideal values). The projection values are shown with their difference from the ideal values given in parentheses. As seen, 61 of the 62 images were correctly identified. We assume

Table III. Worst-case Tests of 100% Perfect Performance 248 Class Set of Four Multi-level (0.33, 0.66, 0.99) Iconic Filters

Input test character	Response for filters $F1$–$F4$			
	$F1$	$F2$	$F3$	$F4$
E	0.3299	0.6601	0.6600	0.6599
T	0.6600	0.3299	0.3301	0.9899
h	0.6599	0.6600	0.9899	0.6599
1	0.9900	0.3299	0.3300	0.9901
6	0.3301	0.3299	0.6599	0.3300

Table IV. Worst-Case Binary Noise Test Results (α = 0.5, σ^2 = 0.25, *Busweek*)

Input test character	Response (and error) for filters $F1$–$F4$			
	$F1$	$F2$	$F3$	$F4$
E	0.24(0.09)	0.55(0.11)	0.62(0.04)	0.75(0.24)*
T	0.57(0.09)	0.28(0.05)	0.29(0.04)	0.91(0.08)
W	0.58(0.08)	0.35(0.02)	0.60(0.06)	1.03(0.04)
h	0.68(0.02)	0.60(0.06)	0.89(0.10)	0.62(0.04)
u	0.62(0.04)	0.85(0.14)	0.92(0.07)	0.90(0.09)
1	0.90(0.09)	0.24(0.09)	0.28(0.05)	0.91(0.08)
3	0.28(0.05)	0.20(0.13)	0.34(0.01)	0.56(0.10)
6	0.27(0.06)	0.29(0.04)	0.57(0.09)	0.30(0.03)
9	0.28(0.05)	0.58(0.08)	0.29(0.04)	0.31(0.02)

Table V. Worst-Case Gray Level Noise Test Results σ^2 = 0.1, *Forbes*)

Input test character	Response (and error) for filters $F1$–$F4$			
	$F1$	$F2$	$F3$	$F4$
Q	0.37(0.04)	0.91(0.08)	0.95(0.04)	0.90(0.09)
R	0.33(0.33)	0.31(0.02)	0.38(0.05)	0.32(0.01)
V	0.58(0.08)	0.31(0.02)	0.62(0.04)	0.70(0.04)
t	1.17(0.18)	0.39(0.06)	0.20(0.13)	0.54(0.12)
u	0.91(0.08)	0.34(0.01)	0.37(0.04)	0.96(0.03)
x	1.03(0.04)	0.34(0.01)	0.60(0.06)	0.91(0.08)
5	0.41(0.08)	0.27(0.06)	0.62(0.04)	0.97(0.02)
6	0.21(0.12)	0.28(0.05)	1.02(0.03)	0.31(0.02)
9	0.38(0.05)	0.68(0.02)	0.35(0.02)	0.32(0.01)
0	0.37(0.04)	0.32(0.01)	0.39(0.06)	0.27(0.06)

projection values with errors below $(\Delta L)/2 = 0.165$ will be correctly thresholded.

Binary noise is typical of the noise expected in OCR applications.[14] We next provided gray-level noise tests. We generated zero-mean Gaussian noise at different variances and added this to each image. We set pixels below 0 to 0 and pixels above 1 to 1, but retained all noise gray levels between 0 and 1. Test were conducted of all 248 images with noise present. The worst-case results for the font *Busweek* are shown in Table V in the same format used in Table IV. As seen, 60 of the 62 images were correctly identified. The gray-level noise used had $\sigma^2_{\text{noise}} = 0.1$. When the noise variance was reduced to $\sigma^2_{\text{noise}} = 0.08$, we obtained 100% correct recognition of all characters. We note that the input SNR is about 31 for $\sigma^2_{\text{noise}} = 0.08$. Figure 5 shows several binary and gray-level noisy input images correctly identified.

We now return to Table II and our theoretical analysis indicating that noise sensitivity and the number of key image pixels increases with N_T. Refer to Table V, which shows that the projection of the letter E on filter $F4$ was 0.75 (in error by 0.24) with $\sigma^2_{\text{noise}} = 0.25$. We reduced the noise threshold to produce noise with $\sigma^2_{\text{noise}} = 0.24$ (only 0.01 different from the prior value). For this noisy image of the letter E we found the projection of the letter E on the fourth filter to be 0.98 (nearly the ideal 0.99 level). Thus with a slightly different noise realization or a slightly different noise level (such that key image pixels were not affected), much larger noise levels can be tolerated. By selecting different projection values for different images and by assigning similar projection codes to similar characters, control over the number of key filter pixels and a reduction in their value is possible.

We now consider tests of these iconic filters with input test images in fonts that were never seen during filter synthesis. Table VI shows the worst case results for tests on input data in the font *Scienam*. As seen, only one error in all 62 characters occurred. Thus properly designed iconic filters can recognize test data that they have never seen. By including fonts of several selected characters, full 100% recognition is possible. The present tests were included to show performance with a limited training set.

Table VI. **Worst-Case New Font (*Sclenam*) Test Results (Error From Ideal Level in Parenthesis)**

Input test character	Font	Response (and error) for filters $F1$–$F4$			
		$F1$	$F2$	$F3$	$F4$
r		0.48(0.18)	0.14(0.15)	0.98(0.01)	0.92(0.07)
S		0.56(0.10)	0.31(0.02)	0.28(0.05)	0.67(0.01)
s	Times	0.97(0.02)	0.30(0.03)	0.30(0.03)	0.36(0.05)
2		0.37(0.04)	0.32(0.01)	0.36(0.05)	1.01(0.02)
4		0.31(0.02)	0.33(0.00)	0.69(0.03)	0.31(0.02)

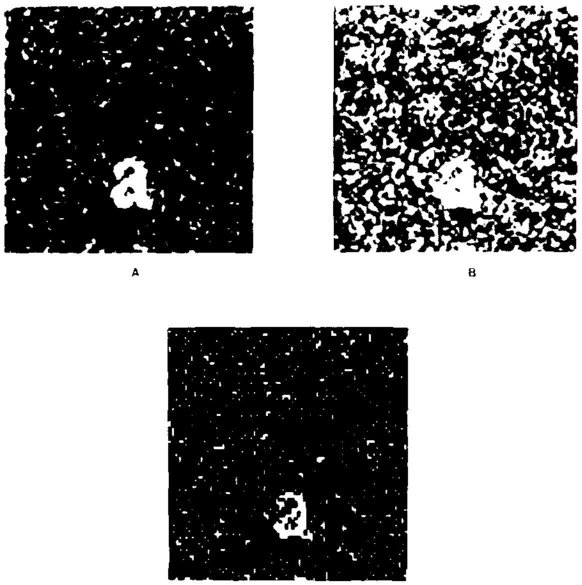


Fig. 5. Typical noisy characters with different noise variances: (a) $\sigma^2 = 0.08$ (gray-level noise); (b) $\sigma^2 = 0.1$ (gray-level noise); (c) $\sigma^2 = 0.24$ (binary noise).

For practical optical realization the dynamic range of the filter function cannot be seven decimal digits as in digital simulations. To quantify the amplitude and phase dynamic range required in the frequency-domain iconic filter, we computed the filters used to digital machine accuracy and then quantized these filters to different numbers of amplitude and phase levels. The worst-case test results were analyzed for the correlation of our multilevel ($L = 3$) multiple ($F = 4$) filters in tests against the 62 characters in the training set font *New York Times* data. These results are typical of those obtained for other fonts. These results showed that a filter quantized in the frequency domain to 32 amplitude levels and 360 (1° resolution) phase levels in the frequency plane performed most excellent, with only two errors out of the 62 characters (96% recognition) with these low quantized filter levels. The use of slightly more amplitude levels and much less phase levels also yielded perfect 100% recognition. Other tests performed considered the uniformity of response of the input spatial light modulator used. These tests showed excellent performance for 5% worst case variation in the spatial uniformity of the input image plane data. We found that up to 30% worst-case nonuniform spatial response in the input device could be tolerated and acceptable results still obtained. Other tests involved rotations of the input object which showed no degradation loss with several degrees of rotation of the input object.

VII. Summary and Conclusion

The issue of large class object recognition has been addressed. New filters for such problems have been described and several hierarchical architectures using them have been discussed. Attention was given to filter synthesis problems foreseen when the number of classes is large. A theoretical basis for the sidelobe and noise performance of such filters was advanced and quantified by experiment. Initial results are quite attractive. Hierarchical correlators and multilevel multiple iconic filters are a viable and attractive solution. They appear preferable to an exhaustive search of all available training images.[15] Training set selection can reduce the number of images necessary and hence clutter. Proper code selection can improve performance and reduce various error sources. Near-perfect recognition of a large number of objects (~1000) with only four filters with moderate filter dynamic range requirements appears possible. Initial OCR tests have quantified these remarks.

The support of this research by the Independent Research and Development Funds of General Dynamics Pomona and by a grant from the Air Force Office of Scientific Research is gratefully acknowledged.

References

1. R. Davis, D. B. Lenat, *Knowledge-Based Systems in Artificial Intelligence* (McGraw-Hill, New York, 1982).
2. R. Davis, B. Buchanan, and E. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program," Artif. Intell. **8**, (1977).
3. T. Kohonen, *Self Organization and Associative Memory* (Springer, New York, 1984).
4. D. Casasent and D. Psaltis, "Deformation-Invariant, Space-Variant Optical Pattern Recognition," Prog. Opt. **16**, 291 (1979).
5. Y. N. Hsu and H. H. Arsenault, "Optical Pattern Recognition Using Circular Harmonic Expansion," Appl. Opt. 21, 4016 (1982).
6. H. J. Caulfield and M. H. Weinberg, "Computer Recognition of 2-D Patterns Using Generalized Matched Filters," Appl. Opt. **21**, 1699 (1982).
7. A. Mahalanobis and D. Casasent, "Large Class Iconic Pattern Recognition: An OCR Case Study," Proc. SPIE **726**, in press, October 1986.
8. R. G. Casey and C. R. Jih, "A Processor-Based OCR System," IBM J. Res. Develop. **27**, 386 (1983).
9. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation," Appl. Opt. **23**, 1620 (1984).

10. D. Casasent and W. T. Chang, "Correlation Synthetic Discriminant Functions," Appl. Opt. **25**, 2343 (1986).
11. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations," Proc. Soc. Photo-Opt. Instrum. Eng. **625**, 220 (1986).
12. D. Casasent and S. A. Liebowitz, "Model-Based Knowledge-Based Optical Processors," Appl. Opt. **26**, 15 May issue (1987).
13. S. A. Liebowitz and D. Casasent, "Error Correction Coding in an Associative Processor," Appl. Opt. **26**, 999 (1987).
14. Y. X. Gu, Q. R. Wang, and S. Y. Suen, "Application of a Multilayer Decision Tree in Computer Recognition of Chinese Characters," *IEEE Trans. PAMI* **5**, 83 (1983).
15. R. R. Kallman, "Construction of Low Noise Optical Correlation," Appl. Opt. **25**, 1032 (1986).

4. OPTICAL STRING CODE OPTICAL PROCESSING

Proc. SPIE
Vol. 848
November 1987

Rule-based String Code Processor

David Casasent and Sung-Il Chien

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

A new and efficient real time technique to produce a string code description of the contour of an object, such as an (angle, length) = (ϕ, s) feature space for the arcs describing the contour, is detailed. We demonstrate the use of such a description for an aircraft identification problem case study. Our (ϕ, s) feature space is modified to include a length string code and a convexity string code. This feature space allows both global and local feature extraction. The local feature extraction follows human techniques and is thus quite suitable for a rule-based processor (as we discuss and demonstrate). Aircraft have generic parts and thus are quite suitable for the model-based description.

1. INTRODUCTION

Aircraft recognition is a classic pattern recognition problem recently surveyed [1]. Many feature spaces have been suggested for such multiple degree of freedom pattern recognition problems. These include: moments [2,3] (which require large dynamic ranges and are noise sensitive when made distortion-invariant); Fourier descriptors [4,5] (which still require feature extraction, computationally intensive matching lists, and which do not lend themselves to use of local information or features); and various curvature features. Our proposed technique handles global and local features, includes feature extraction with in-plane distortion-invariance and avoids a large matching search.

We selected a string code description of the object. Other work with similar descriptions [6-9] has also been used and their VLSI realization discussed [10-12]. However, our string code description (ϕ, s) = (angle, length) of the arcs on the contour of an object is generated most efficiently and allows global and local feature space analysis. Global features are necessary for general problems and local features allow specific problems to be solved quite effectively. The local features we use correspond to specific object parts and thus allow rule-based analysis (since this is the manner in which humans achieve identification). Our edge description is different from the conventional chain code [9] and we do not convert the chain code to an (x, y) or other description as others [7] do early in the processing period. Our rule-based technique differs from syntactic [13] techniques. Our rule-base follows a forward chaining control flow as does SPAM [14]. As our model knowledge, we employ specific aircraft structural and part information.

Section 2 describes our case study, model base, and data base. Section 3 provides an introduction and overview of our processor and our feature space. Section 4 details our new efficient feature space generation technique and includes typical results. Section 5 briefly discusses our rule-based processor.

2. DATA BASE

The case study we consider is the identification and orientation estimation of 10 different aircraft. Fig.1 shows the top-down views of these aircraft grouped by the functional role of the aircraft. In our tests, all aircraft are 128 x 128 pixels in resolution. Our model base contains different polygon descriptions of all aircraft and their parts, from which any aspect view can be produced quite easily [15].

3. PREPROCESSOR OVERVIEW

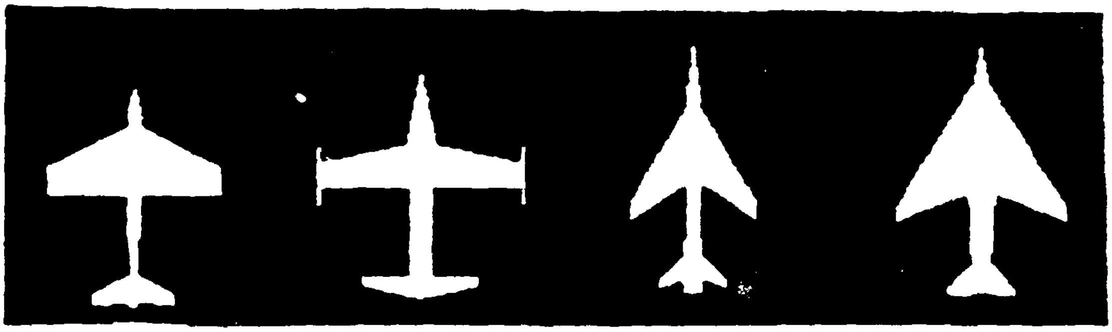
Our full processor contains five major sections as shown in Fig.2. The preprocessor performs edge enhancement (this is necessary to produce good peaks in the Hough transform space we will employ) and generates a clockwise ordered list of pixel coordinates for the contour or boundary of the object (using classic techniques [16,17]). The feature space produced is a (ϕ, s) description of the angle (ϕ) and the length (s) of all arcs clockwise in a string code connected object boundary or contour description. An aspect estimator unit determines if the aircraft is being viewed nearly top-down or if an out-of-plane distorted image is being investigated. A rule-based or an associative processor are used (depending upon the aircraft object's distortions). In this present paper, we discuss the rule-based processor. Thus, in this initial work, we will restrict attention to nearly top-down aircraft views.

4. EFFICIENT (ϕ, s) STRING CODE FEATURE SPACE GENERATION

The first step is to reduce the clockwise ordered contour pixel list to N (approximately 20-30) vertices. Fig.3 shows a DC10 (Fig.3.a) and its boundary description with the vertices noted (Fig.3.b). The N vertices define N arcs for the boundary, each with a length (s) and an internal angle (ϕ). Fig.3.c defines the angle ϕ. The result is a (ϕ, s) string code.

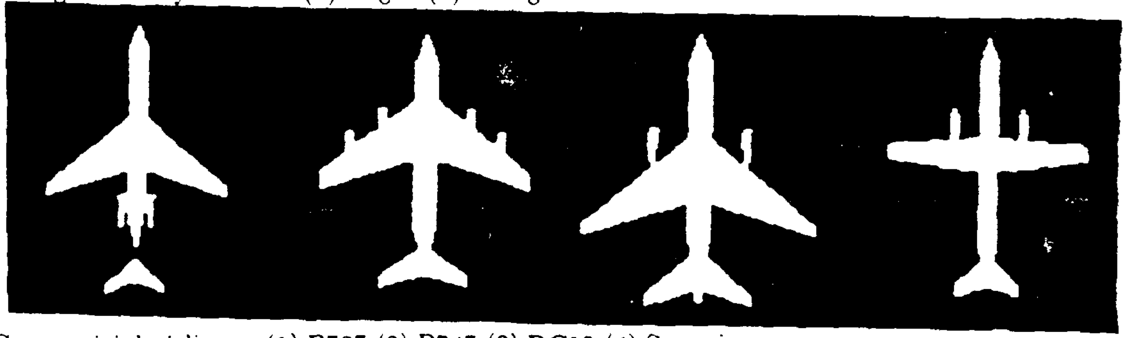
The block diagram of our efficient (ϕ, s) string-code generation system is shown in Fig.4. We use the clockwise-ordered contour list of the boundary pixels (x, y), form the Hough transform (HT) of the input from the original data, and locate the six major (and true) HT peaks and their (p, θ) values. We then Hough transform each contour pixel and check if it evokes a peak at one of the (p, θ) six major HT peak parameter locations. This assigns most contour points to the six major lines in the image and gives automatically (without time-consuming trigonometric operation) the angle ϕ and the length (s) of these lines. Only a small fraction of the pixel points in the contour list remain to be assigned ϕ and s values. Each of these is a connected set of pixels that lies in a gap between previously assigned points. We achieve the (ϕ, s) description of these pixels into lines by a conventional split-line fitting method [18,19]. This split-line technique is computationally expensive, but (with the six major lines and our HT technique) this needs only to be applied to a significantly reduced number of points in the contour list. Thus, this technique generates the full (ϕ, s) string code description quite efficiently.

A HT converts lines in the input into points in a (p, θ) parameter Hough space, i.e. a. coordinates corresponding to the normal distance (p) and the angle (θ with respect to the x axis) of the normal of the line, with six peak heights proportional to the number of points on the line (or the length of the



(a) U.S.A. military aircraft: (1) B57 (2) F104 (3) F105 (4) Phantom

(b) Foreign military aircraft: (1) Mig21 (2) Mirage



(c) Commericial airliners: (1) B727 (2) B747 (3) DC10 (4) Swearingen

Figure 1: Image Data Base (128 x 128)

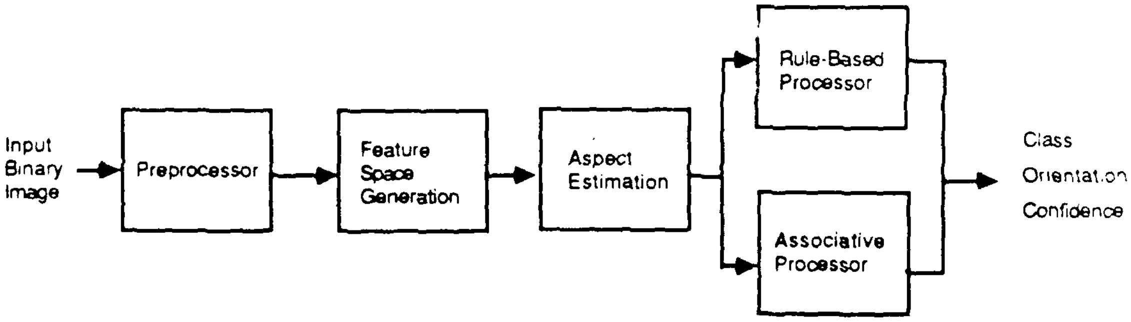
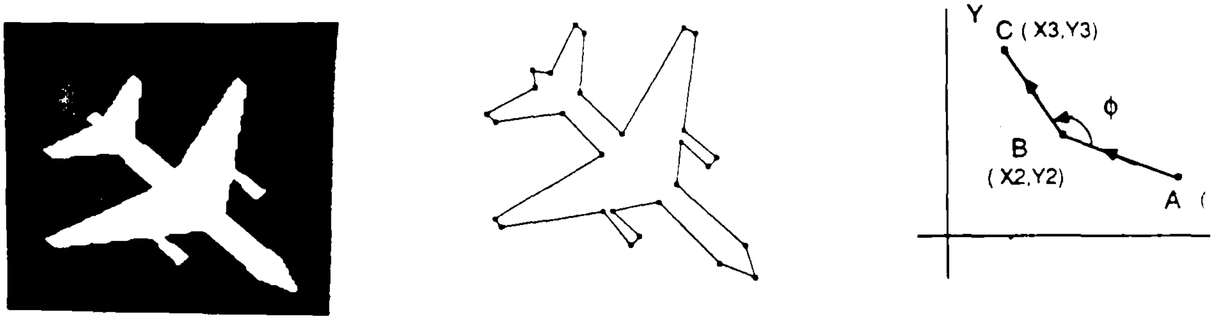


Figure 2: Overall Processor



(a) DC10 (b) DC10 vertices (c) Angle ϕ Definition

Figure 3: Example of vertices describing an object boundary (Fig.3 a and b) as arcs of length s and internal angle ϕ (ϕ is defined in Fig. 3.c)

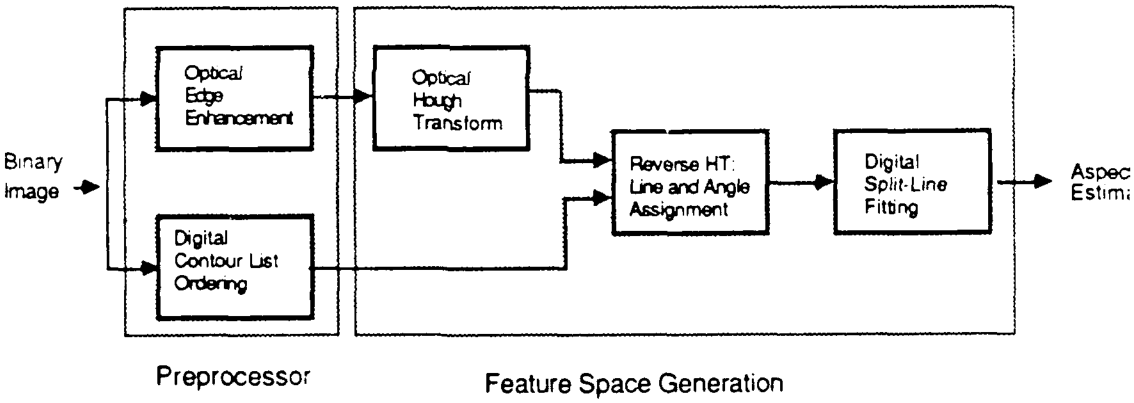


Figure 4: Block diagram of an efficient (ϕ, s) string code processor

line). Fig.5.a shows the HT for the DC10 with the nose vertical. Fig.5.b shows the HT for th with the nose horizontal. The two major peaks in Fig.5.a lie on the $\theta = 0°$ line and in Fig.5.b on the $\theta = 90°$ line. These two major peaks denote the presence of the fuselage and its orienta Fig.5, we see six major peaks, however this does not always occur (when noise, quantization image resolution, and 3-D roll and pitch distortions occur). To demonstrate this and techni overcome these problems, we show (in Table 1) the 10 largest HT peaks obtained for th oriented at 120°. This demonstrates specifically that the largest six HT peaks do not corres the major lines in the image, specifically HT peak 6 and 7 are false peaks that are larger than pe (which is the next largest true peak). We note [20] that such false peaks occur close to the true] (within three pixels for our aircraft data). Thus, we employ an algorithm that ignores HT s peaks that lie within four pixels of the large peak. Employing this rule, the six proper p corresponding to the six major lines in the aircraft image emerged (Table 2). Table 3 lists the aircraft lines corresponding to the six major HT peaks and Fig.6.a shows the lines in the air image itself. Fig.6.b shows the resultant final (ϕ, s) image with all vertices obtained (including t obtained by the split-line fitting technique).

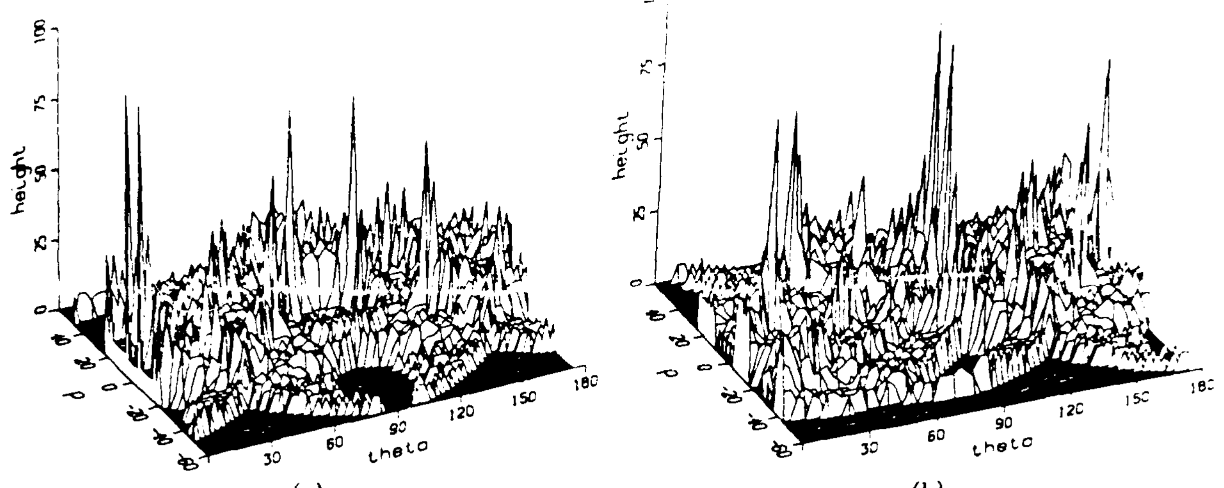
An efficient technique to assign the θ and p parameters of the six HT peaks to point in the con list is now detailed. To achieve this, we transform each pixel coordinates (x, y) in the clock contour list into a sinusoid. This sinusoid needs only be evaluated at the six θ values of the dominant HT peaks and at the p coordinates within each. Thus, these HT operations on the con list are easily achieved. Since we expect a number of successive pixels in the contour list (those each arc) to correspond to the same HT peak point, the processor can be quite fast (and very effici compared to typical techniques involving extensive trigonometric calculation).

We now discuss the descriptions we employ of the string code representation of the object symbolic descriptor. We first consider the full (ϕ, s) string code with the exact analog values fo angles and lengths. Next, we consider a convexity string code. This lists only the convexity of angles of the arcs in the boundary representation as convex V (if $\phi < 180^\circ$) or concave C (if ϕ 180°). Last, we consider a length string code which lists only the length of each arc as : very sh short, medium, long, and very long. These are expressed in terms of maximum differ $\Delta = L_{max} - L_{min}$ in the length L of the arcs for the input image. Each length region is $\Delta/6$ ex for the medium length region which is $\Delta/3$ in extent. These different symbolic string descriptions of the object contour are found to be quite useful for global and local rule-b processing, as described in Section 5.

5. RULE-BASED PROCESSOR

Our rule-based system employs if-then rules, a context-limited and rule-ordered control stra and forward chaining with five rule groups used as we now describe. The first rule group (star rules) locates the fuselage.

The second rule group concerns substructure search rules. The purpose of this second rule grou to locate all separate regions of an object and to divide them into left (L) and right (R) regions respect to the fuselage. We first extract the fuselage and all vertices corresponding to it. separates the contour list into L and R regions. We group these into separate connected reg (closed polygon boundaries) corresponding to parts of the object. For each such region, we calcu its area, perimeter, compactness, and its position with respect to the fuselage. Various rules are to determine the type of each region. Three representative examples are given below:



(a) (b)

Figure 5: HT of DC10 with nose oriented vertical (a) and horizontal (b)

Hough Peak	p(pixel)	θ(degree)	Peak Heigh
1	3	165	10
2	-19	114	9
3	-5	60	9
4	19	6	9
5	5	60	8
6	-20	111	7
7	20	9	6
8	3	135	6
9	-7	63	5
10	-5	162	5

Table 1: Data on the 10 largest peaks for a DC10 with its nose at 120 °

Hough Peak	p(pixel)	θ(degree)	Peak Heig
1	3	165	1
2	-19	114	9
3	-5	60	9
4	19	6	9
5	5	60	8
6	3	135	6

Table 2: Data on the six largest HT peaks using our false peak algorithm. The six peaks noted are the correct ones.

Corresponding Aircraft Part
Right Line on Fuselage
Left Line on Fuselage
Right Front Wing Line
Right Rear Wing Line
Left Front Wing Line
Left Rear Wing Line

Table 3: 6 major lines in an aircraft

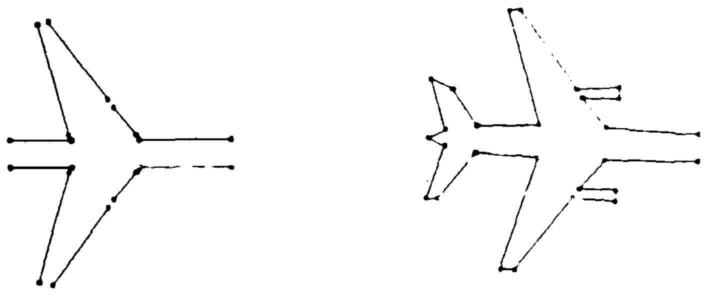

(a) (b)

Figure 6: Aircraft Image with
(a) only the six major arcs and (b) all arcs

> Rule 1: Wings are the largest regions in L and R. They must have the proper spatial relationship to the fuselage.
>
> Rule 2: If the convexity symbolic code for a region has all vertices convex, then this region is a wing with no engines etc on it.
>
> Rule 3: If the convexity symbolic code for a region has two concave vertices out of four adjacent vertices and if this correspond to short arcs, then this region is a wing with an engine etc on it.

From the location of the concave vertices and arcs of short length, the position of the engine et (refered to as a "blob") or small structure on the wing (or fuselage) can be determined. We discus this further below. Fig.7 shows examples of a wing region with no engine (Fig.7.b) as detected from it convexity code (Fig.7.a). Fig.8 shows an analogous example when the convexity code (Fig.8.a) show several C sections and hence indicates the presence of an engine in the image of Fig.8.b. Followin such rules, we can segment the L and R regions into parts as shown in Fig.9 (wings, tails, and blobs).

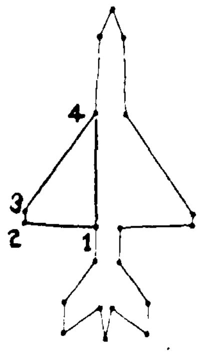
The third rule group we use provides a check on the top-down orientation estimation (this obtained from the number of regions in L and R, the areas of these regions, and the symmetry of th L and R sections), yaw estimates (these are obtained from the θ coordinate of the fuselage peak in th HT space), and roll estimates (from the symmetry or ratios of areas in regions L and R).

The fourth rule group concerns substructure rules. These are intended to identify the small or loc features or object regions or parts. The best example of this concerns "blobs" on wings an specifically whether these are engines, missiles, or fuel tanks. For the image data base we considere we note (from Fig.1) that if the blobs appear in the center of the wing, the blob is an engine (e. DC10); and if it appears on the tip of a wing, it is a missile (e.g. F104).

The fifth rule group contains classification rules. We note three examples below. There ar approximately 40 rules used in total.The following are intended to be representative examples. Befor

Vertex	Convexity Code
1	**V**
2	**V**
3	**V**
4	**V**

(a)

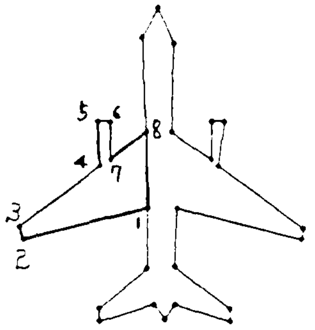


(b)

Figure 7: Example of a convexity code (a) for a wing region with no engine (b)

Vertex	Convexity Code
1	**V**
2	**V**
3	**V**
4	**C**
5	**V**
6	**V**
7	**C**
8	**V**

(a)



(b)

Figure 8: Example of a convexity code (a) for a wing region with an engine on it (b)

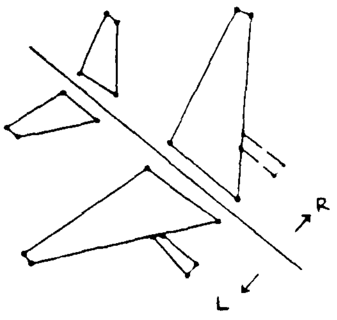


Figure 9: Representative left (L) and right (R) segmented regions of an aircraft

discussing these, we note one additional parameter included in our feature space parameters the angles ϕ_1 and ϕ_2 that the wings make with the fuselage at points A and B (see Fig.10).

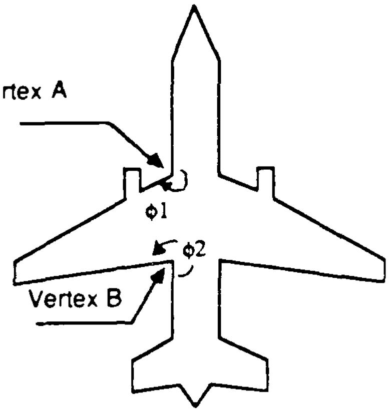


Figure 10: Definition of the internal angles ϕ_1 and ϕ_2 at vertex points A and B in an aircraft

Using these blob and angle parameters, we note three rules as examples:

Rule 1: If a blob is present on a wing, and if it is an engine (i.e. in the center of the wing), and if the angle ϕ_1 at vertex A (Fig.10) $> 245°$, then the aircraft is a Swearingen.

Rule 2: If a blob is present on a wing, and if it is an engine, and if the angle ϕ_1 at vertex A $\leq 245°$, then the aircraft is a DC10.

Rule 3: If a blob is present on a wing, and if it is not an engine (i.e. it exists at the tip of the wing), then the aircraft is an F104.

Comparison of the Fig.1 images and these rules shows that these rules correctly classify these aircraft noted.

6. SUMMARY AND CONCLUSION

We have advanced an efficient HT technique to assign lengths and angles of most arcs to a clockwise pixel coordinate list of the contour or boundary points. This is complemented by a split-line fitting algorithm which need be applied only to small gaps in the residual boundary. For the case study of an aircraft data base (which is very suitable for model-based description), we separate the object into L and R regions, each described by connected polygons, each of which are identified as wings, tails, fuselage, engines, etc. Convexity and length symbolic string codes aid this separatior This feature space is most efficiently obtained and it allows us to apply both global features (suitabl for general pattern recognition) and local features (necessary to handle distorted objects and parti images). The local features used correspond to specific object points (easily obtained and described i our symbolic notation) that humans also relates to. The feature space and case study considere (aircraft identification) lends itself naturally to a rule-based processor. Examples of rules and their us in the identification of aircraft classes were provided. The general technique is the most flexible. Whe augmented with an associative processor, the potential of the system is even further increased.

ACKNOWLEDGEMENTS

The support of this research by a grant from the Air Force Office of Scientific Research on Optica Symbolic Processing is gratefully acknowleged.

REFERENCES

1. J.F. Gilmore, "Air Targeting of The Third Kind: Airborne Vehicles", SPIE proc., Applications of Digital Image Processing 7,Vol. 504, 1984, pp. 330-340.

2. S.D. Dudani, K. Breeding, and R. McGhee, "Aircraft Identification by Moment Invariants", IEEE Trans. on Computers,Vol. C-26, No. 1, January 1977, pp. 39-45.

3. F.A. Sadjadi, and E.L. Hall, "Three Dimensional Moment Invariants", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 2, No. 2, March 1980, pp. 127-136.

4. T. Wallace and P. Wintz, "Algorithm Using Normalized Fourier Descriptor", Computer Graphics and Image Processing, Vol. 13, No. 5, 1977, pp. 99-126.

5. O.R. Mitchell, and T.A. Grogan, "Global and Partial Shape Discrimination for Computer Vision", Optical Engineering, Vol. 23, No. 5, Sept./Oct. 1984, pp. 484-491.

6. T. Wallace, "Local and Global Shape Description of Two- and Three-dimensional Objects", PhD dissertation, Elec. Eng., Purdue University, 1979

7. T. Wallace, O.R. Mitchell, K. Fukunaga, "Three-Dimensional Shape Analysis Using Local Shape Descriptors", Proc. IEEE Pattern Recogntion and Image Proc. Conf., August 1979.

8. T. Wallace, O.R. Mitchell, K. Fukunaga, "Three-Dimensional Shape Analysis Using Local Shape Descriptors", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 3, May 1981, pp. 310-323.

9. H. Freeman, "Computer Processing of Line Drawing Images", Computer Surveys, Vol. 6, No. 1, March 1974, pp. 57-98.

10. R. M. Lea, "VLSI and WSI Associative String Processors for Structured Data Processing", IEE Proceedings, Vol. 113, No. 3, June 1986, pp. 113-124.

11. C.A. Finnila, and H.H. Love, "The Associative Linear Array Processor", IEEE Trans. on Computers,Vol. C-26, 1977, pp. 112-124.

12. R.M. Lea, "The Comparative Cost of Associative Memory", Radio & Electron. Eng.,Vol. 46, No. 10, Oct. 1976, pp. 487-496.

13. K.C. You, and K. S. Fu, "Syntactic Shape Recognition Using Attributed Grammars", Purdue University Technical Report, TE-EE 78-38, August 1978.

14. D.M. McKeon, W.A. Harvey, and J. McDermott, "Rule-Based Interpretation of Aerial Imagery", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 7, Sept. 1982, pp. 570-585.

15. D. Casasent, and S. Liebowitz, "Model-Based Knowledge-based Optical Processors", Applied Optics,Vol. 26, No. 10, May 1987, pp. 1935-1942.

16. D.H. Ballard and C.M. Brown, Computer Vision, Prentice Hall, 1982, pp. 143-145.

17. T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, 1977, pp. 142-159.

18. R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, Inc, 1973, pp. 338-339.

19. D.H. Ballard and C.M. Brown, Computer Vision, Prentice Hall, 1982, pp. 232-235.

20. C.M. Brown, "Inherent Bias and Noise in the Hough Transform", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 5, No. 5, Sept. 1981, pp. 493-505.

5. REAL TIME LIQUID CRYSTAL TELEVISION FEATURE SPACE GENERATION

Reprinted from Applied Optics, Vol. 26, Page 938, March 1, 1987
Copyright © 1987 by the Optical Society of America and reprinted by permission of the copyright owner.

Real-time deformation invariant optical pattern recognition using coordinate transformations

David Casasent, Shao-Feng Xia, Andrew J. Lee, and Jian-Zhong Song

The well-known scale and rotation invariant polar-logarithmic coordinate transformation is used to achiev in-plane distortion invariant pattern recognition. The coordinate transform is produced by a computer generated hologram on a laser printer. Attention is given to weighting terms in the output and their effect o resolution and the number of input plane pixels removed near the origin. The optically produced coordinat transformed input pattern is interfaced to a correlator by a pocket liquid crystal TV to provide real-tim processing. Experimental results are included.

I. Introduction

Optical pattern recognition using a matched spatial filter and a correlator is a well-known technique.[1] It is advantageous due to its high speed and parallel processing. But the conventional correlator cannot recognize scaled or rotated images of the reference object. For example, for a 1% scale change of the reference object, the SNR of the resultant correlation peak can be 10 dB down from that of the autocorrelation, and a 20-dB loss can occur for a 1.7° rotation of the input from the reference.[2] This disadvantage limits the potential applications of the conventional correlator. One solution to these problems is development of a space variant optical processor which is realized by applying a coordinate transformation preprocessing operation to the input and reference data.[3] Coordinate transformations, such as the logarithmic transformation (which results in a Mellin transformation, which is scale invariant), the polar ($r - \theta$) transformation (which results in rotation invariance), and the combination of the two[3] (the ln$r - \theta$ coordinate transformation, which results in scale and rotation invariance), have been reported.

Here we report the optical implementation of deformation invariant real-time optical pattern recognition using a computer-generated hologram (CGH) and a liquid crystal television (LCTV). The CGH is used with a Fourier transform lens to perform the ln$r -$ coordinate transformation. The use of a hologram consisting of many interferometrically produced holographic optical elements (HOEs) for coordinate transforms has been demonstrated.[4] The principle of using a CGH for a coordinate transformation was demonstrated earlier for the Mellin transform[5] and for the circle-to-point[6] and ln$r - \theta$ transformations.[7] A discussion of the fabrication of our CGH is presented i Sec. II together with several issues associated with th optical coordinate transformation and their effects o our real-time correlator. The LCTV and a TV camer are used to connect the coordinate transform preprocessing system to a conventional optical matched spatial filter correlator in real time. The LCTV introduces a phase distortion in the wavefronts passin through it which has been corrected using a phas conjugate filter.[8] Real-time scale and rotation invariant pattern recognition is demonstrated experimentally in Sec. III. Our conclusions are advanced in Sec. IV

II. Design of the Coordinate Transformation CGH

The system to achieve the ln$r - \theta$ coordinate transformation is shown in Fig. 1. The input $f(x,y)$ is place in contact with a continuous phase CGH with transmittance $h(x,y) = \exp[j\phi(x,y)]$, where $\phi(x,y)$ is th phase distribution of the phase filter. Lens L_1 form the Fourier transform of the product $f(x,y)h(x,y)$ a the plane P_1, where we find

$$F(u,v) = \iint f(x,y)\exp[j\phi(x,y)] \times \exp[-j(2\pi/\lambda f_L)(xu + yv)]dxdy. \qquad (1$$

where λ is the wavelength of the laser used, and f_L is th focal length of lens L_1. For the ln$r - \theta$ coordinat transformation, we desire

---

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Pittsburgh, Pennsylvania 15213.

Received 12 September 1986.
0003-6935/87/050938-05$02.00/0.
© 1987 Optical Society of America.

$$u(x,y) = \ln(x^2+y^2)^{1/2} = \ln r,$$
$$v(x,y) = -\tan^{-1}(y/x), \qquad (2)$$

and the integral in Eq. (1) can be solved using the approximate saddle point integration method.[9] For the coordinate transform in Eq. (2), a continuous phase solution $\phi(x,y)$ exists since $u(x,y)$ and $v(x,y)$ have continuous partial derivatives and since the partial derivatives of u with respect to y and of v with respect to x are equal. The desired phase function is

$$\phi(x,y) = (2\pi/\lambda f_L)[x\ \ln(x^2+y^2)^{1/2} - y\tan^{-1}(y/x) - x]. \qquad (3)$$

A. CGH Design

There are several techniques that may be used to form the desired phase filter.[10] Since the amplitude transmittance of $h(x,y)$ is one, we need only record the phase function, and since this is recorded by positioning the data on the mask, binary CGH recording techniques can be used. Since a continuous phase function solution exists, we thus use a binary computer-generated interferogram[11] for the CGH. The interferogram is the interference pattern of $\phi(x,y)$ and a plane wave reference at an angle θ. The maxima of this interference pattern (the locations of the interference fringes or the lines that must be plotted on the CGH) must satisfy

$$2\pi\alpha x - \phi(x,y) = 2\pi n, \qquad (4)$$

where n is an integer which denotes different fringes and where the carrier frequency $\alpha = (\sin\theta)/\lambda$. The recorded CGH is generally photoreduced onto film, and Eq. (4) describes the final CGH. To avoid overlapping between the first-order and second-order diffracted waves in the diffraction plane P_1, α must satisfy[11]

$$\alpha > (1.5/\pi)\max\left|\frac{\partial\phi(x,y)}{\partial x}\right|. \qquad (5)$$

Inserting Eq. (3) into (5) with $x_{\max}$ and $y_{\max}$ being the maximum size of the input image or the CGH, we obtain

$$\alpha > (3/\lambda f_L)\ln(x_{\max}^2 + y_{\max}^2)^{1/2}. \qquad (6)$$

This result has not previously been given *full* attention and is of concern since it affects resolution, as we discuss in Sec. II.C. We note that we detect only the first-order diffraction pattern at P_1. In the experiments that we performed, we used the parameters $x_{\max} = 5$ mm, $y_{\max} = 5$ mm, $\lambda = 0.6328\ \mu$m, and $f_L = 400$ mm. From Eq. (6), we then find $\alpha > 23$ line pairs/mm is required. We used $n = 400$ fringes in Eq. (4) for $\alpha = 40$ line pairs/mm. We solved for the various (x,y) that satisfy Eq. (4) for each value of n, connected these points, plotted the associated lines on an Imagen 300 laser printer, and then photoreduced the plot to the final CGH size of 10 × 10 mm.

B. Space Bandwidth Product Requirements

This $\ln r - \theta$ input image representation space (that is scale and rotation invariant) is detected by a TV

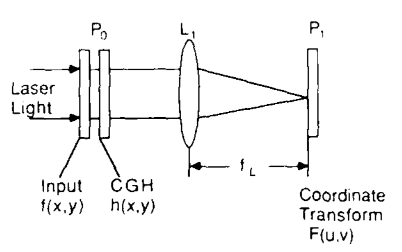


Fig. 1. Schematic of optical coordinate transformation system.

camera at P_1 of Fig. 1, and the electronic output from the TV camera is then fed to an LCTV in the input plane of an optical matched spatial filter frequency plane correlator. We now relate the space bandwidth product ($N_r \times N_\theta$) required in the $\ln r - \theta$ space at P_1 to the input image space bandwidth product $N \times N = N^2$ at P_0. The radial Δr and angular $\Delta\theta$ spatial sampling increments are both $\sqrt{2}/N$, i.e., a factor of $\sqrt{2}$ larger than *the reciprocal of the number of input samples N.* Including the effect of the number of samples M omitted near the origin of the input image pattern, we find

$$N_r = N\ln(N/M)/\sqrt{2}, \qquad N_\theta = (4N/\sqrt{2})\tan^{-1}(N/2). \qquad (7$$

These results follow from others[3] extended to the case of an $\ln r - \theta$ transform. The 2-D space bandwidth product required at P_1 to sample adequately the $\ln r -$ plane is thus

$$N_r N_\theta = 2N^2\ln(N/M)\tan^{-1}(N/2) = \pi N^2\ln(N/M). \qquad (8$$

where the final result follows for large N.

C. Intensity Weighting Effects

To evaluate Eq. (8), we must select M. To do this, we consider the weighting present at P_1 and then obtain a new criteria for selection of M and hence the P_1 resolution required for a given input P_0 resolution. The intensity of each transformed point (u_a,v_a) in the P output $F(u,v)$ is[10]

$$|F(u_a,v_a)|^2 = |4\pi^2 f^2(x_a,y_a)/(\phi_{xx}\phi_{yy} - \phi_{xy}^2|$$
$$= |\lambda^2 f_L^2 f^2(x_a,y_a)(x^2+y^2)|, \qquad (9$$

where ϕ_{mn} denotes the partial derivative of $\phi(x,y)$ wit respect to m and n and where (x_a,y_a) is the input poin in P_0 that contributes to the output point (u_a,v_a) in P From Eq. (9), we see that the P_1 pattern associate with a given input P_0 point depends on the intensity o each input point and its position in P_0. Our concern i the effect of the positional weighting factor given b the square radius $r^2 = (x^2 + y^2)$ of each input point i P_0. The effect of the $r^2 = (x^2 + y^2)$ weighting factor best described for the case of an input $f(x,y)$ pattern o uniform intensity. In this case, points further fro the optic axis in P_0 will be brightest in the coordinat transform pattern at P_1, and points near the center o P_0 (near $r = 0$) will be the dimmest in P_1. This attractive since these points must be omitted in the l coordinate transform. Tapering of the input illum nating light can conceptually correct this effect (e cept near $r = 0$, which is not of concern since this regi is blocked). Without correction for this effect, a scaled input image will result in the same shaped P_1 pattern but with a different intensity (larger intensity if the input object is larger). When this transformed pattern is used in a correlator, the r^2 weighting is of no concern, since the matched filter would also include the same r^2 weighting.

Our present concern with the r^2 weighting term in Eq. (9) is it effects on M and the size of each diffraction order in P_1. Points near $r = 0$ in P_0 map to high frequencies, and these frequencies approach infinity for P_0 points approaching $r = 0$. Thus separation of diffraction orders at P_1 becomes impossible and requires an increasing α unless M points near $r = 0$ are omitted at P_0. The α calculations in Eqs. (5) and (6) considered such issues but do not readily allow one to select M. Fortunately, the transform intensity of the points near $r = 0$ is so weak due to the r^2 attenuation factor in Eq. (9) that they can be ignored, and thus P_1 diffraction orders of finite size result. If we assume that plane P_1 intensities for which the weighting factor in Eq. (9) is <1% of the maximum can be omitted, we find that this corresponds to $N/M = 10$ in Eq. (8). The space bandwidth product N_rN_θ at P_1 is now related to that of the input (N^2) by $(N_rN_\theta) = 7.2N^2$. In our system, the coordinate transformed image at P_1 is fed into the LCTV, which has a square resolution of 120 × 120. For this output P_1 space bandwidth, the resolution that our CGH can accommodate is ~40 × 40. The choice of M affects the amount of scale change that the system can accommodate,[3] but our $N/M = 10$ choice is sufficient for a large range of scale.

Another issue of potential concern is the intensity of the output from a correlator with P_1 as an input. When the input image rotates, the transformed output image is cyclically displaced along the vertical axis at P_1. In a correlator, this can result in two correlation peaks rather than one. The intensity of the two peaks will sum to the intensity of the single autocorrelation peak, and one peak will always be at least 50% of the intensity of the autocorrelation peak. This effect can be avoided by synthesizing a CGH and the matched spatial filter to cover a rotation range from 0 to 4π rather than 0 to 2π. For the case of a scaled input, the P_1 pattern shifts horizontally depending on the scale factor and intensity of the pattern increases for scale increases. A correlation output threshold set based on the minimum scale expected (this also affects the choice of M) should thus be used (or different correlation plane thresholds can be used for different vertical correlation plane coordinates). Alternatively, from the dc value of the Fourier transform of the coordinate transform of the input, an estimate of the energy of the object is available and can be used to set an adaptive correlation threshold.

III. Real-Time Deformation Invariant Correlation Results

The CGH was tested in the system of Fig. 1 with various input aircraft and letter images. The output P_1 pattern in Fig. 1 was seen to remain unchanged (except for shifts) for rotations and scale changes in the P_0 input images. The P_1 output transformed pattern was found to shift horizontally by $\ln a$ for input scale changes a and to shift cyclically vertically proportional to input rotations. This verified the use of the CGH for the desired $\ln r - \theta$ coordinate transform.

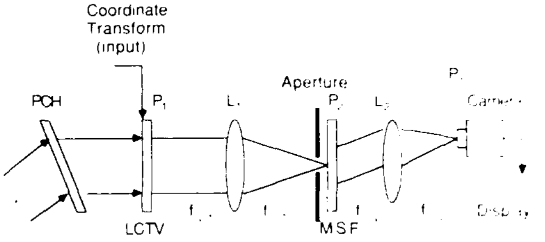


Fig. 2. Real-time optical correlator system schematic.

To perform deformation-invariant optical pattern recognition in real time, a spatial light modulator such as the LCTV is required to record the input P_0 pattern and often also the coordinate transformed pattern at P_1 of Fig. 1. If the P_1 data are used as a feature space, the system is modified slightly[12] to provide a shift invariant P_1 output which can then be detected and fed to a feature extractor and classifier. In this paper, we concern ourself with the case when the P_1 data are fed to the input of a correlator (as shown in Fig. 2). In this case a device such as an LCTV is required to contain the P_1 data from the system of Fig. 1. We achieved this by feeding the TV detected output of the P_1 pattern of Fig. 1 to an LCTV at P_1 of Fig. 2. The phase errors of the LCTV are corrected for by the phase conjugate hologram (PCH) shown.[8] A matched spatial filter of the coordinate transformed object to be recognized is formed at P_2 with the beam balance ratio chosen to yield the optimal correlation SNR. The output correlation is produced at P_3, where it is detected by a camera and displayed on an isometric display. The aperture at P_2 passes only the first-order diffracted pattern from P_1. (Several diffracted orders exist due to the regular pattern of pixels on the LCTV.) This removes the effect of the fixed LCTV pattern and improves the SNR of the output correlation obtained.[8] The video output from the camera in P_3 is amplified and partly saturated to improve the output display and reduce the r^2 weighting factor in Eq. (9).

The results of our real-time experiments on the systems of Figs. 1 and 2 demonstrating scale and rotation invariant pattern recognition are now discussed. Figure 3 demonstrates rotation invariance. Figure 3(a) shows the original input image used, the letter X, and Fig. 3(b) shows the autocorrelation of its coordinate transformed pattern with the peak in the center of the P_3 correlation plane. The size of the input characters was ~50% of the input field of view with an equivalent resolution of ~20 × 20 pixels in P_1 of Fig. 1. Figure 3(c) shows the $\ln r - \theta$ coordinate transform of Fig. 3(a). This was used to synthesize the matched spatial filter at P_2 of Fig. 2. Figures 3(d) and (e) show the isometric displays of the P_3 output correlation plane for 30° rotations of the input image clockwise [Fig. 3(d)] and

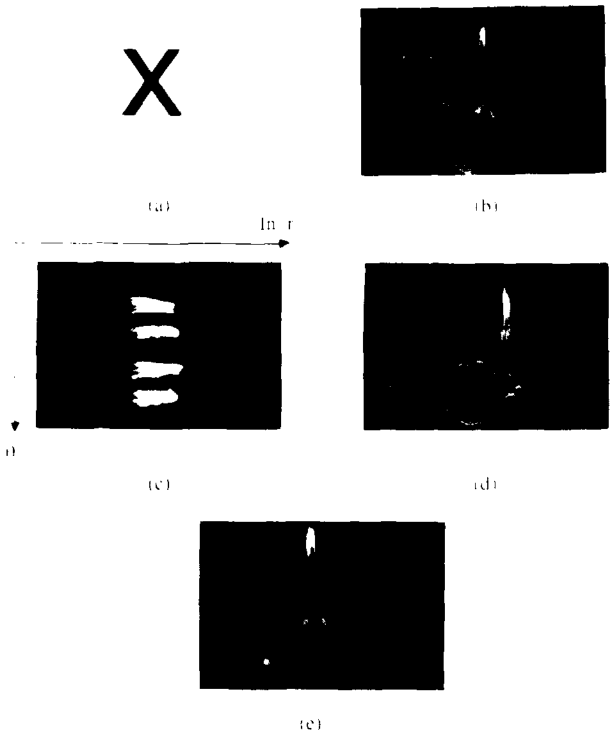


Fig. 3. Real-time laboratory rotation invariant object recognition data: (a) input object; (b) the autocorrelation of the coordinate transformed pattern; (c) the coordinate transformed pattern; (d) correlation for an input object rotated clockwise by 30°, (e) correlation for an input object rotated counterclockwise by 30°.

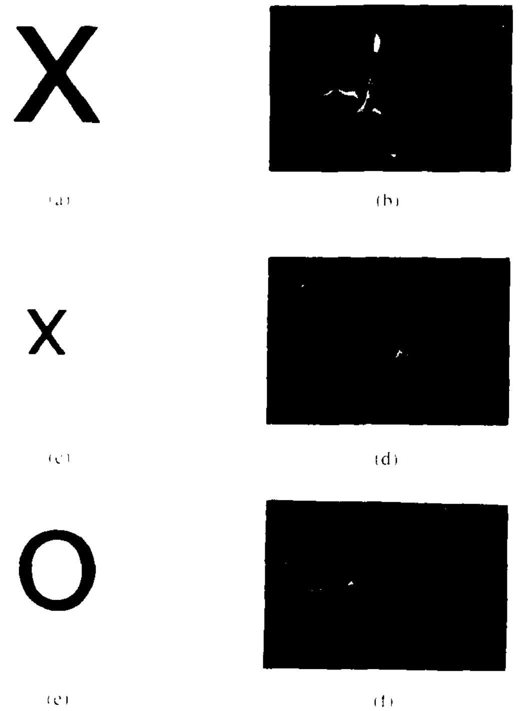


Fig. 4. Real-time laboratory scale invariant object recognition and cross-correlation data: (a) input object, 130% scale change from reference; (b) correlation of the coordinate transformed input of (a); (c) input object, 70% scale change from reference; (d) correlation of the coordinate transformed input of (c); (e) input object different from the reference object; (f) correlation of the coordinate transformed input of (e).

counterclockwise [Fig. 3(e)], respectively. These figures show a large correlation peak whose shape and peak value are quite constant. This indicates the occurrence of the reference object in the input image. The output correlations clearly demonstrate that the correlation peak is maintained under input rotations and that it is displaced up and down proportional to the rotations of the input pattern.

The scale invariance of our real-time system is demonstrated in Fig. 4. The same original image and matched spatial filter were used [Fig. 3(a)]. Its coordinate transformed pattern [Fig. 3(c)] and autocorrelation [Fig. 3(b)] were shown earlier. Scaled versions of the reference input, as shown in Figs. 4(a) and 4(c), with scale factors of 1.3 and 0.7, respectively, were used as inputs. Figures 4(b) and (d) show the isometric displays of the corresponding output correlation planes. Note in these figures that the correlation peaks are still largely unchanged in shape and are now displaced in the horizontal direction from that of the autocorrelation in Fig. 3(b) proportional to the logarithm of the scale change of the input. We note also that the value of the correlation peak varies for a scale change of the input as expected since a larger pattern (containing more energy) results in more energy in the output plane. The cross correlation of an unknown input image [Fig. 4(e)] with the matched filter of the coordinate transformed reference yields negligible output [Fig. 4(f)] as expected, since the coordinate transformation is one-to-one and thus does not make cross-correlation response larger.

IV. Conclusions

The use of an optical coordinate transform (CT) system, employing a CGH and a lens, in series with a conventional optical correlator has been demonstrated in real time for in-plane deformation invariant pattern recognition. The CT system is interfaced to the correlator system using a LCTV and TV camera to allow the system to process data in real time.

In our system, the CT chosen performed the polar-$\ln r$ transform which yields scale and rotation invariance. The CGH used to perform this transformation was detailed with attention to the recording technique, space bandwidth required, and effects of an r^2 weighting term. The scale and rotation invariant real-time correlation performance of our system was experimentally demonstrated. The results using the inexpensive LCTV are promising, and the use of higher-resolution LCTVs should yield improved correlator performance at modest expense.

The support of this research by the Air Force Office of Scientific Research and the Jet Propulsion Laboratory is gratefully acknowledged as is use of the facilities of the Center for Excellence in Optical Data Processing at Carnegie Mellon University.

Shao-Feng Xia is on leave from Fudan University, China, and Jian-Zhong Song is on leave from Changchun Institute of Optics & Fine Mechanics, China.

References

1. A. VanderLugt, "Signal Detection by Complex Spatial Filtering," IEEE Trans. Inf. Theory **IT-10**, 139 (1964).
2. D. Casasent and A. Furman, "Sources of Correlation Degradation," Appl. Opt. **16**, 1652 (1977).
3. D. Casasent and D. Psaltis, "Deformation Invariant, Space Variant Optical Pattern Recognition," Prog. Opt. **16**, 289 (1978).
4. H. Bartelt and S. Case, "Coordinate Transformations via Multifacet Holographic Optical Elements," Opt. Eng. **22**, 497 (1983).
5. D. Casasent and C. Szczutkowski, "Optical Mellin Transforms Using Computer Generated Holograms," Opt. Commun. **19**, 217 (1976).
6. J. Cederquist and A. Tai, "Computer Generated Holograms for Geometric Transformations," Appl. Opt. **23**, 3099 (1984).
7. Y. Saito, S. Komatsu, and H. Ohzu, "Scale and Rotation Invariant Real Time Optical Correlator Using Computer Generated Hologram," Opt. Commun. **47**, 8 (1983).
8. D. Casasent and Shao-Feng Xia, "Phase Correction of Light Modulators," Opt. Lett. **11**, 398 (1986).
9. M. Born and E. Wolf, *Principles of Optics* (Pergamon, New York, 1965).
10. O. Bryngdahl, "Geometrical Transformations in Optics," J. Opt. Soc. Am. **64**, 1092 (1974).
11. W. H. Lee, "Computer Generated Holograms: Techniques and Applications," Prog. Opt. **16**, 119 (1978).
12. D. Casasent and A. J. Lee, "A Feature Space Rule Based Optical Relational Graph Processor," Proc. Soc. Photo-Opt. Instrum. Eng. **625**, No. 33, 234 (Jan. 1986); see also D. P. Casasent and A. J. Lee, "Optical Relational-Graph Rule Based Processor for Structural Attribute Knowledge Bases," Appl. Opt. **25**, 3065 (1986).

6. REAL TIME OPTICAL COMPUTER GENERATED HOLOGRAM LASER PRINTER REALIZATION

Reprinted from Applied Optics, Vol. *26*, Page 136, January 1, 1987
Copyright © 1987 by the Optical Society of America and reprinted by permission of the copyright owner.

Computer generated hologram recording using a laser printer

Andrew J. Lee and David P. Casasent

The use of a laser printer for recording various types of computer generated holograms is discussed. a
results are presented.

Computer generated holograms (CGHs) have a variety of uses in optical information processing.[1-6] Many CGH recording devices can be used,[7-10] but few are inexpensive and easily available to the researcher first hand. Recording with a Calcomp plotter and subsequent photographic reduction of the pattern is the most accessible form of CGH recorder. However, it is limited in its flexibility, resolution, and reproducibility, and it requires photographic reduction of large 20- × 20-in.[2] patterns. The advent of laser printers and their reduced costs makes them attractive CGH recorders. We emphasize the use of the Imagen 300 laser printer,[11] although the same techniques apply to other laser printers.

The Imagen 300 is commonly used to print letters and other documents using word processing software. In this mode, the word processor generates a file written in imPRESS code. This file is fed to an image processor (IP) within the Imagen printer which stores and interprets this file and converts it to a raster. This raster format is necessary to control sequentially the writing laser beam. The print engine within the Imagen contains the laser and optics which perform the printing of the information on paper as a high resolution binary pattern. The imPRESS commands typically used define English and Greek characters, fonts, and symbols. To employ the device for CGHs, the user can employ imPRESS to define his own fonts by defining glyphs, tne basic cells used in halftone printing of greyscale imagery. The user can also employ imPRESS commands that draw points, lines, and arcs and perform area shading. We now detail two procedures we have developed to use the Imagen printer fc
synthesis. We also quantify accuracy measur
taken on the printer.

There are a large variety of CGH encodin
niques possible to produce grey-scale and cc
value data with binary recording devices such
printers. To record a 2-D rectangular array
with different transmittance at each point, tl
array is specified, and halftone techniques (usi
defined glyphs or the shading command in im
are used to produce the desired transmittance
point. For spatial filtering and matched spatia
ing applications, other encoding techniques[1
possible but follow from the above basic techni
pictorial example of a halftone encoded ima
duced by the Imagen printer is shown in Fi
demonstrate the results and concept. The ima
sists of 190 × 190 glyphs which encode 64 differe
levels.

For more general CGHs, the required patte
sists of a set of curves, each described by an e
and (for the case of a binary pattern) one must
all the points satisfying each equation and prod
resultant plot of these curves. The imPRESS co
DRAW-PATH draws a curve through a number of
A file of all these DRAW-PATH commands (one 1
curve) and the absolute pixel locations of pc
each are then produced. This is referred tc
graphic imPRESS file. This file is then sent
Imagen printer where the IP interprets the ir
commands and produces a raster file which
which pixels on the page should be turned on (
black). The print engine then produces the s
pattern. An example of such an output is sl
Fig. 2. This is a continuous phase binary s
CGH that implements a polar-log coordinate t
mation on a 2-D input image. This CGH is us
necessary for space-variant scale and rotation
ant pattern recognition.[15]

One issue in implementing these concepts
absolute pixel positions must be used rather tl

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Pittsburgh, Pennsylvania 15213.

Received 9 June 1986

0003-6935/87/010136-03$02.00/0.

© 1986 Optical Society of America.

Fig. 1. Grey scale image produced on the Imagen laser printer as an example of a CGH with halftone grey level spatial transmittance

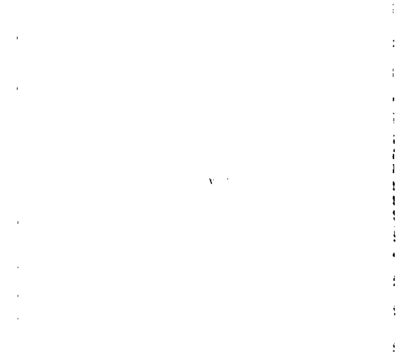


Fig. 2. Imagen laser printer produced continuous phase b synthetic CGH that achieves a polar-log coordinate tran mation.

ventional units (inches) of distance. This issue arises, since when one uses a CGH in an optical system, its physical size must be calculated and specified, and the CGH must be produced to exactly this size. Another issue is that the imPRESS commands are written as hex character pairs, and pixel locations are written as four hex pairs. This introduces some difficulty in use and debugging if the user is unfamiliar with hex representation and with the hex description of all imPRESS commands (since to read an imPRESS file, all hex characters must be converted to their decimal or command equivalents). The first technique we use to generate the graphic imPRESS file is to write FORTRAN subroutines that set $(x = 0, y = 0)$ at a given absolute pixel position and then convert all (x,y) pairs from distance units to pixel values (by dividing by the 300-pixel/in. resolution of the Imagen printer). The result is an (x,y) sampling at 300 pixels/in. We have found this technique to be the most accurate, although it is the most difficult to use and debug. The second technique we use to generate the imPRESS file uses DISSPLA[16] graphics software called from a simple FORTRAN program. The points (x,y) to be connected are left in inches (or any distance unit), or as pixel indices, and are then connected via a series of CONNPT commands. The software then converts these DISSPLA commands into imPRESS commands and the (x,y) points into pixel indices. This technique is much easier to use since the user need not know all imPRESS commands or their hex equivalents and how to convert from inches to pixel indices to hex characters. However, each pixel on the printed page is not separately controlled, and pixel points are not always placed in an exact desired position (due to sampling and interpolation deficiencies in the DISSPLA software). DISSPLA also produces automatic margins, thus eliminating many possible points on the edge of the page and hence reducing the total number of pixels one can record. We now quantify many of these above remarks. These two techniques and the Imagen system are shown in the block diagram flow chart of Fig. 3.

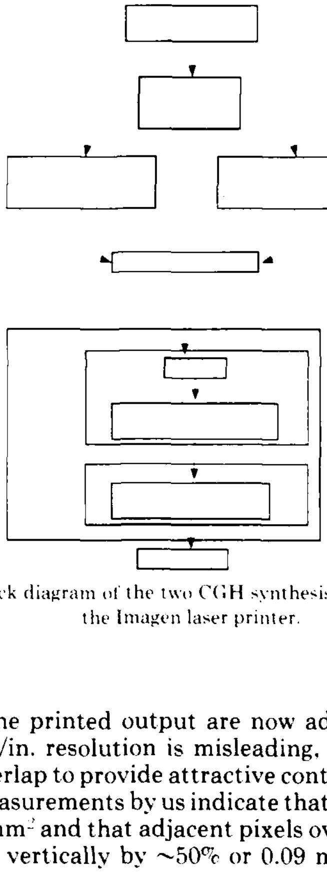


Fig. 3. Block diagram of the two CGH synthesis techniques the Imagen laser printer.

The pixel size, overlap of pixels, and positional accuracy of the printed output are now addressed. 300-pixel/in. resolution is misleading, since adja pixels overlap to provide attractive continuous cha ters. Measurements by us indicate that a pixel is (× 0.175 mm² and that adjacent pixels overlap hori tally and vertically by ~50% or 0.09 mm. Thus center-to-center spacing of adjacent pixels is (mm, and each pixel is 0.175 mm in size in one di sion. As a result, the sequence of three pixels a OFF, ON will not show a central OFF pixel. Thus printer resolution is 150 nonoverlapping pixel However, each pixel location can be specified to part in 300/in. There is a slight variation in the w of pixels due to the varying density of the toner. variation is quite small [(below several microns) a random and could not be measured with our avai techniques, even after 20× magnification]. The absolute positional reproducibility of points was tested by writing alternate pixels and lines on the left and right side of a page and on the top and bottom of a page. In all cases, straight lines resulted that were aligned exactly to the desired pixel position. Thus the Imagen printer used with the imPRESS commands is reproducible within the specified pixel resolution and within excellent measurement accuracy limitations. To calibrate distances to pixels and to quantify the absolute positional accuracy, the outline of a square 1200 × 1200 pixels in size was recorded using the imPRESS commands. The two dimensions of the resultant plot were measured to be equal within 0.5 mm (0.02 in.) or 3 pixels. Thus the absolute positioning accuracy of the printer is 3/1200 or 0.25% over a distance of 4 in. It is important to note that the spatial size of the square CGH pattern was precise (i.e., 4 in. within 0.02 in., corresponding to 1200 pixels) when written directly by the imPRESS commands. When the same 4- × 4-in. square (1200 × 1200 pixels) was written using DISSPLA software to generate the imPRESS file, the size of the square produced was 3.76 in. This is due to sampling and interpolation effects in the DISSPLA software (whose source code is not available). This represents no major problem, since one simply scales the desired dimensions by 4/3.76 to obtain an exact pattern size. The DISSPLA software is still not capable of controlling each pixel on the final printed page and in the final imPRESS file. To demonstrate this, we wrote a pattern of two ON pixels separated by 1/300 in., 2/300 in., etc. and found the imPRESS file generated to have scaling errors in the number of OFF pixels. Thus, for best absolute accuracy with separate direct control of each image pixel, the imPRESS software is recommended directly. However, for most CGHs with moderate resolution, the more user friendly DISSPLA software synthesis technique will suffice.

The final topic of concern is the number of points that one can record. The IP within the Imagen printer produces the necessary raster image from the imPRESS file. In conventional text writing, the imPRESS commands used do not involve lines that extend more than a fraction of an inch. Thus the printer engine can (and does) start printing before an entire page raster file has been produced in the IP. In recording various CGHs, the last command in the imPRESS file can involve points separated by a considerable distance (in the extreme case, a command to draw a line from the top to the bottom of the page). Thus, in CGH synthesis, the entire image raster file must be complete before printing begins. We achieve this with a special software command that stops the print engine until this condi-

tion is satisfied. For conventional text recordi
command is not used, since it considerably redu
printing speed possible. The standard IP h
kbytes of memory for storage and processin
have added an additional several megabytes of
ry to this to accommodate high resolution lar
CGH synthesis. For an 8 × 10 = 80-in^2 printin
the printer can support 80 (300)^2 = 7.2-M pixel
CGHs. The need for a large memory is thus im
in CGH synthesis.

CGHs are increasing in use and popularity
ease with which they can be produced on inex
and generally available laser printers should
CGH techniques to more researchers.

References

1. O. Bryngdahl, "Optical Map Transformations," Opt. C
10, 164 (1974).
2. D. Casasent and C. Szczutkowski, "Optical Mellin Tra
Using Computer Generated Holograms," Opt. Commur
(1976).
3. S. H. Lee, Ed., *International Conference on Compute
ated Holography*, Proc. Soc. Photo-Opt. Instrum. E
(1983).
4. J. Cederquist and A. Tai, "Computer Generated Holog
Geometric Transformations," Appl. Opt. **23**, 3099 (198
5. D. Casasent and J. Song, "A Computer Generated Holo
Diffraction Pattern Sampling," Proc. Soc. Photo-Opt.
Eng. **523**, (1985).
6. D. Casasent, "Computer Generated Holograms in Patt
ognition: A Review," Opt. Eng. **24**, 724 (Sept./Oct. 19
7. H. J. Caulfield, "National Computer Holography
Opens," *Laser Focus* (Nov. 1982), pp. 42–46.
8. S. M. Arnold, "Electron Beam Fabrication of Compute
ated Holograms," Opt. Eng. **24**, 803 (Sept./Oct. 1985).
9. R. A. Athale, C. L. Giles, and J. A. Blodgett, "Use of
Written CGH in Optical Pattern Recognition," Proc. S
to-Opt. Instrum. Eng. **437**, 48 (1983).
10. R. Sandstrom and S. H. Lee, "Production of Optical Co
Transform Filters by a Computer Controlled Scanning
ometric Pattern System," Proc. Soc. Photo-Opt. Instr
437, 64 (1983).
11. Imagen Corp., *imPRESS Programmer's Manual* (265
mas Expressway, P.O. Box 58101, Santa Clara, CA 950
12. W. H. Lee, "Sampled Fourier Transform Hologram G
by Computer," Appl. Opt. **9**, 639 (1970).
13. C. B. Burckhardt, "A Simplification of Lee's Method
ating Holograms by Computer," Appl. Opt. **9**, 1949 (1
14. J. P. Allebach and J. J. Keegan, "Computer Synthesis
Fourier Transform Holograms Using Ordered Dither,
Soc. Am. **68**, 1440 (1978).
15. D. Casasent and D. Psaltis, "New Optical Transform
tern Recognition," Proc. IEEE **65**, 77 (1977).
16. Integrated Software Systems Corp., *DISSPLA User's
(ISSCO, Inc., 10505 Sorrento Valley Road, San Diego,
1985).

7. OPTICAL ERROR CORRECTING ASSOCIATIVE PROCESSORS

Reprinted from Applied Optics, Vol. *26*, Page 999, March 15, 1987
Copyright © 1987 by the Optical Society of America and reprinted by permission of the copyright owner.

Error-correction coding in an associative processor

Suzanne Liebowitz and David Casasent

A technique for encoding binary outputs from optical filters or matrix memories used in an assoc
processor for object recognition is discussed. Binary coded output vectors (rather than unit vectors) are
and considerably improve storage capacity. The output codes or matrix memories are chosen from
theory to enable error correction and detection. The error classification rate for the coded sche
compared to the noncoded version for different amounts of noise in the input and output planes. Discu
of extensions to more classes, more errors, and multilevel coding are included.

I. Introduction

We describe a technique for using conventional coding theory to enhance the capability of optical correlators for object recognition and orientation determination. Three types of advanced filter that have been suggested for use in an optical correlator are projection filters,[1] correlation filters,[2] and peak-to-sidelobe ratio (PSR) filters.[3] Section II reviews the synthesis of these filters. Here, we emphasize the use of projection filters and especially their ability to encode multiple-class information. Several methods for implementing associative memories have been detailed in the literature.[4-13] Some of these methods have been proposed for optical implementation.[7-13] In this work, we synthesize the associative memory from projection filters. A recent suggested method of optical associative memory synthesis used projection filters to form a matrix with each filter as a column and optically computed the vector inner products required in parallel.[13] The output vector from this memory is a code that describes the input vector. In our work, the input vector is an object image-plane representation, and the output code indicates the class of the object. We will use the term class loosely, since each input image can either be a different object or a different orientation of one object (or a combination of both). Each bit in the output code corresponds to the output of one of several filters (matrix columns). This associative memory
formulation is based on a multifilter classific
technique. In Sec. II we review its formulation
realization and note its improved storage capacit

In this paper we further enhance this metho
designing the filters and the output codes to e
error detection and correction. For our work, w
binary coding theory since it allows for easier co
tions/encodings. Therefore, the outputs of the fi
can only be set to 0 or 1 (or values representing 0 o
for example, a 0 output should be avoided). The e
correcting technique used in our work is present
Sec. III. Error-correcting codes are advantageo
situations where the probability of a bit transitio
a binary code, this is the probability that a
incorrect) is small. In Sec. IV we discuss the
model used in our work and the theoretical limita
of the coding techniques. In Sec. V we presen
results of simulations tested on both uncoded
coded outputs with a data base consisting of le
from the alphabet. By limiting the scheme to a bi
code, we lose the ability to handle more classes
fewer filters as is possible with multilevel coding
Sec. VI we will discuss how multilevel coding c
used to enhance the capability of the system to h
more classes with fewer filters and other selecte
vanced considerations.

II. Filter Synthesis and Associative Memory Formul

The conventional heteroassociative memory fo
lation uses unit output vectors with the location
denoting the recollection vector or class assoc
with the input data. Various associative memory
thesis techniques and realization architectures
been described.[4-13] We consider an efficient,
capacity multifilter associative memory. A si
conventional optical associative processor is sho
Fig. 1. It has an input key vector **x** at P_1, whi
multiplied by the associative memory matrix **M**

The authors are with Carnegie-Mellon University, Department of Electrical & Computer Engineering, Pittsburgh, Pennsylvania 15213.

Received 12 September 1986.
0003-6935/87/060999-08$02.00/0.
© 1987 Optical Society of America.

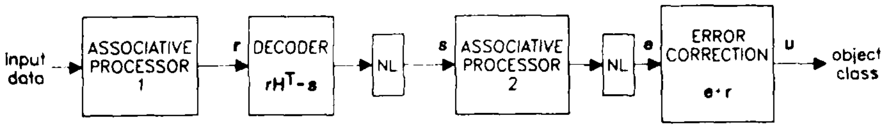


Fig. 1. Optical parallel r
tion of multifilter coding (a
tive memory scheme)

to give an output recollection vector $y = \mathbf{M}\,\mathbf{x}$ at P_3. In our associative memory synthesis, we use $k = F$ filters $\mathbf{h}_k$ as the columns of $\mathbf{M}$. The P_3 output vector thus has F elements, each of which is the vector inner product of the $\mathbf{x}$ input and the various $\mathbf{h}_k$ filters. For the case of $F = 2$ filters ($\mathbf{h}_1$ and $\mathbf{h}_2$) at P_2 with binary thresholded P_3 outputs, the four possible F = 2-bit output vectors are noted in Table I. Each of these can be made to correspond to a different object class by the appropriate output binary encoding. For the general case of F filters (F columns in the matrix at P_2), the F-bit output can accommodate 2_F classes of objects (it is often preferable to allow $2^F - 1$ output classes to avoid the all-zero output vector, which can also occur with no P_1 input). The associative memory matrix $\mathbf{M}$ need only be $M \times F$, where M is the dimension of the input vector. Conventional associative memories require far larger matrix sizes and would provide at most recognition of F rather than 2^F output classes (while also requiring $F \ll M$ for most conventional associative memory formulations).

Synthesis of this matrix (and its associated filter vectors or columns) has been well-documented[1,3,13] and is thus only briefly highlighted here. We begin with several images in each of several classes and form their vector inner product matrix $\mathbf{V}$. We then invert $\mathbf{V}$ and multiply it by a matrix $\mathbf{P}$ whose rows are the desired F-digit output $\mathbf{y}_k$ recollection vector codes for each input key vector $\mathbf{x}_k$. The rows of the resultant matrix $\mathbf{A} = \mathbf{V}^{-1}\mathbf{P}$ specify each filter function $\mathbf{h}_k$ as a linear combination of all the original key vectors. The F filters $\mathbf{h}_k$ are then used as the columns of the matrix $\mathbf{M}$ at P_2 of Fig. 1 and the F-digit $\mathbf{y}$ output at P_3 will be the binary code for the 2^F different object classes desired and specified by the $\mathbf{P}$ matrix. The more general version of this associative memory synthesis algorithm uses F filters with L different levels allowed in each of the different F output P_3 digits. This allows us to represent L^F object classes with an associative memory with only F column vectors. We will restrict attention here to the case of binary output vectors (because the error-correcting techniques we will be describing will be much simpler for this case). In this paper, we consider techniques to improve the performance of such associative processors by using coding theory to allow the detection and correction of digit errors in the output $\mathbf{y}$ vector. We will also restrict attention to projection filters (with extensions to correlation
other advanced filters[3] following directly). Whe
key vectors are chosen properly (as statistically r
sentative of the data),[3] this associative processor
forms quite well and the output vector denote
reference $\mathbf{y}_k$ recollection vector most closely assoc
with the $\mathbf{x}$ test vector. The matrix can also be sy
sized to output a reference key vector $\mathbf{x}_k$ most cl
associated with a partial or noisy input key vector
is an autoassociative memory matrix). In this p
we will consider only a heteroassociative memory
trix, although an autoassociative memory matrix
mulation as well as the cascade of an autoassoci
and a heteroassociative memory matrix is possible
yields excellent results. Our main attention wi
given to providing error-correction ability to this
ciative processor in addition to the initial error-co
tion ability the system possesses as an associ
memory and/or nearest-neighbor processor. Suc
ditional error correction is necessary when parti
put vectors are present, when the dynamic range o
optical processor implementing the memory is lo
when input or output noise is large. The basic e
correction techniques advanced should be suitab
most associative processor synthesis algorithms
architectural realizations.

Table I. Two-Filter Output

Class	$\mathbf{h}_1$	$\mathbf{h}_2$
1	0	0
2	0	1
3	1	1
4	1	0

III. Error-Correction Coding

The basic idea of coding theory is to represent a
output by $n > k$ bits in order to allow for error co
tion. A simple example of a binary coding tech
which adds redundant bits is the parity bit schen
which one extra bit is added to a k-bit representa
The extra bit indicates if the number of ones i
code is odd or even. This technique helps dete
rors but cannot correct them. For our present ap
tions, it is desirable to use a coding scheme tha
allow for correction. The choice of coding schen
use for this problem is exhaustive. Many (k +
codes exist that can allow recognition of 2^k objects
various abilities to detect and correct errors. A v
of decoding schemes also exist. The group of cod
chose to investigate is the linear block codes.
have the ability to correct errors. The more bit
that a code is able to correct, the more redundan
one needs in the representation. Linear block c
are described in terms of generator matrices $\mathbf{G}$, p
check matrices $\mathbf{H}^T$, and a syndrome vector $\mathbf{s}$. An
linear code uses n bits to represent a k-bit code
our binary case) 2^k different objects where $n > k$

To demonstrate the concept, we specifically ch
use a (7,4) Hamming code. A Hamming code is
able because it involves a matrix–vector multipli
for decoding (and this operation can be implem
digitally or optically using a nonlinearity su

Table II. Decoding Table for the $(n,k) = (7,4)$ Hamming Code

Syndrome s	Bit in error	Coset Leader e
000	0	0000000
100	1	1000000
010	2	0100000
001	3	0010000
110	4	0001000
011	5	0000100
111	6	0000010
101	7	0000001

thresholding). A (7,4) Hamming code uses 7 bits to encode 4-bit data. It can thus accommodate $2^4 = 16$ different inputs or classes, and the code has $7 - 4 = 3$ redundant bits. This particular code can detect and correct 1-bit error in the output. We now provide a brief review of conventional Hamming code theory.[14] The n-bit code is derived by multiplying each possible k-bit message by a $k \times n$ matrix **G** known as the generator matrix. The (7,4) Hamming code is derived by multiplying each 4-bit message **u** (i.e., 0000, 0001, . . . or 1111) by

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

In coding theory, vectors are row vectors ($\mathbf{u}^T$ is a column vector), a matrix–vector multiplication is written as **u G,** and all multiplications are modulo 2. We will retain this notation and usage. For the message **u** = [1101], the n-bit code word would be **u G** = [0001101]. For example, the first element of **u G** is

$$[1101][1011]^T = (1 + 0 + 0 + 1)_2 = 2_2 = 0. \tag{2}$$

The **G** matrix can be written as an augmented matrix **G** = [**P**|**I**], where **I** is a $k \times k$ identity matrix (here $k = 4$) and **P** is a $k \times (n - k) = 4 \times 3$ matrix with 0 and 1 values chosen for the specific code.

To decode a received message **r** (of n bits) to produce the original k-bit message, we multiply **r** by a parity-check matrix $\mathbf{H}^T$, where $\mathbf{H} = [\mathbf{I}_{n-k}\ \mathbf{P}^T]$. In our example, **I** is $n - k = 7–4$ or is a 3×3 identity matrix, **H** is 3×7, and $\mathbf{H}^T$ is 7×3. The product $\mathbf{r}\ \mathbf{H}^T$ yields a syndrome vector of dimension $n - k = 3$ for our example. Note that this $\mathbf{r}\ \mathbf{H}^T$ multiplication is also modulo 2. The syndrome vector tells us if an error has occurred in transmission and which bit is in error. If $\mathbf{s} = \mathbf{O}$ (the zero vector), no error has occurred. A nonzero vector **s** indicates the presence of an error as well as which of the n bits in the received message is in error. Table II shows the eight possible 3-bit syndrome vectors for our $(n,k) = (7,4)$ code example, the associated bit that is in error, and an n = 7-bit unit vector **e** called a coset leader. The location of the 1 in **e** indicates which bit in the received message is in error. The corrected received code is obtained by adding **e** to **r** modulo 2 (with no carries). This correction operation can be performed by a bit-by-bit exclusive-OR of **e** with **r**.

The relationship between **e** and **s** is usually implemented in a lookup table. We propose to achieve this

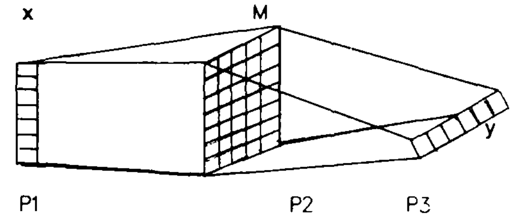


Fig. 2. General block diagram of an error-correcting associa
processor.

with an associative memory to determine the sy drome-coset leader association. Since each syndro vector $\mathbf{s}_i$ corresponds to a coset leader $\mathbf{e}_i$, we will p duce an $\mathbf{s}_i$ for each input $\mathbf{e}_i$ by a matrix–vector multip cation by a matrix **Y** that satisfies $\mathbf{s}_i\mathbf{Y} = \mathbf{e}_i$ for all ($\mathbf{s}_i$ pairs. If we place each $\mathbf{s}_i$ in the ith row of a matri *and each* $\mathbf{e}_i$ *in the* i*th row of a matrix* **E,** *then* **Y** specified by

$$\mathbf{SY} = \mathbf{E}.$$

Equation (3) can be solved in several ways[4]: in a lea squares sense as $\mathbf{Y} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{E}$, or iteratively, or fr the outer-product approximation (assuming ortho nal vectors $\mathbf{s}_i$ such that $\mathbf{S}^{-1} = \mathbf{S}^T$)

$$\mathbf{Y} = \sum_i \mathbf{s}_i{}^T\mathbf{e}_i.$$

Figure 2 shows the general block diagram of proposed error-correcting associative processor. T n = 7-bit Hamming-coded received vector **r** is outp from the first associative processor (Fig. 1 with the matrix synthesized using projection filters). It is th decoded by multiplication with the parity-check n trix $\mathbf{H}^T$. The $n - k = 3$-bit syndrome vector **s** p duced is then converted to the coset leader vector **e** the second associative processor shown (which has t same form as Fig. 1 with a different P_2 matrix). T presence of a bit error and which bit (if any) is in er is determined by **e**. The final box then produces t corrected n-bit **u** code. The vector operations p formed are modulo-2 (no carries) and, thus, the se rate operations cannot be combined conventiona into one matrix–vector processor. However, they be combined into one table lookup associative proc sor (Fig. 2). However, the use of additional nonline ity appears to be beneficial in such processors, and employ the system in the form shown in Fig. 2. emphasize the nonlinear nature of the various ope tions, we include nonlinear (NL) units in Fig. 1. Th nonlinear units also include thresholding operation reduce noise effects and to improve performance.

Let us now discuss how **s** and **e** provide error corr tion. Three situations can occur for the **s** output any Hamming code. We discuss these for our case an $(n,k) = (7,4)$ code:

(1) The received vector is one of the sixteen allo ble n = 7-bit codes **uG** for the k = 4-bit words **u**. this case, **s** and **e** will be zero. The received code w

r will be correct and the final n-bit word **u** will be correct.

(2) The received vector **r** has a 1-bit error. In this case, **s** will be one of the $2^{n-k} - 1 = 7$ nonzero syndrome vectors and **e** will denote which bit is in error (see Table II). In this case, **e** and **r** can always correct the error to yield the correct **u**.

(3) The received vector **r** has more than 1-bit error. In this case, the vector will be (incorrectly) corrected to one of the sixteen Hamming code words. This is because the Hamming code is designed such that each of the sixteen 7-bit received codes has seven 7-bit received code words that have 1 bit in error. The seven codes which are 1-bit different are unique to each of the sixteen code words. Therefore, the $16 \times 7 = 112$ seven-bit codes will always be corrected to one of the sixteen error-free Hamming code words. Including the original sixteen Hamming code words, $112 + 16 = 128$ (all 2^7) possibilities for 7-bit outputs are accounted for.

Further details on Hamming codes and other linear block codes are provided in many texts.[14-17] Other coding schemes can allow the presence of more than 1-bit error to be detected and, therefore, provide a no decision output state possibility.

IV. Output Probability of Error and Noise Model

Coding techniques perform well if the probability p of a bit error is small. In this section we derive the amount of noise that the coding scheme can tolerate and still be effective. In our specific Hamming code example, the probability of error p for any bit in the noncoded 4-bit representation is

$$P_1(e) = 1 - (1 - p)^4. \qquad (4)$$

The probability of error for 7-bit Hamming code is the probability that two or more bit errors occur or

$$P_2(e) = 1 - (1 - p)^7 - 7p(1 - p)^6. \qquad (5)$$

which is 1 minus the probability that none or 1-bit errors occur. If p is small, then we can use the series expansion $(1 - x)^n$ to approximate (4) and (5) by

$$P_1(e) = 1 - (1 - 4p) = 4p, \qquad (6)$$

$$P_2(e) = 1 - (1 - 7p + 21p^2) - 7p(1 - 6p) = 21p^2. \qquad (7)$$

The approximation used in (6) and (7) holds if $P_1(e) > P_2(e)$, i.e., if

$$p < 4/21 \text{ or } p < 0.2. \qquad (8)$$

Therefore, for small p, Eq. (6) is greater than Eq. (7) and we expect an advantage in using the coding methods to correct errors. If p is large (i.e., if the noise is large), coding may not be beneficial.

We now derive a first-order estimate of the amount of noise our system can tolerate and still provide error-correction ability. We model the noise fed to the output of the system as a Gaussian zero-mean variable n. The noise n is generated and added to the output c of the filter to produce $c' = n + c$. This is tl thresholded at +0.5 and the pixels or elements of received signal become 0 if $c' < 0.5$ *and* 1 *otherw* This is the output **r** of our noisy system. The variai σ^2 of the additive noise is related to p as we now det From (8), we require $p < 0.2$ to satisfy our approxir tions in (6) and (7). We assume that the probabil that any bit is a 1 (or a 0) is 0.5. Therefore, if an out element of the noiseless system is 1, it will become 0 < -0.5; similarly, if an output element of the noisel system is 0, it will become 1 if $n > 0.5$. For Gauss noise, the probability of a bit transition error is thu

$$p = 0.5(1/2\pi)^{1/2} \int_{-\infty}^{-0.5} \exp(-x^2/2\sigma^2)dx$$

$$+ 0.5\,(1/2\pi)^{1/2} \int_{0.5}^{\infty} \exp(-x^2/2\sigma^2)dx.$$

We denote the Gaussian distribution as

$$G(x) = (1/2\pi)^{1/2} \int_{-\infty}^{x} \exp(-t^2/2)dt,$$

and note that $G(-x) = 1 - G(x)$. Using this symmet property, Eq. (9) becomes

$$p = 1 - G(0.5/\sigma).$$

For this to be <0.2, we require $G(0.5/\sigma) > 0.8$ or

$$\sigma < 0.6 \text{ or } \sigma^2 < 0.36.$$

Therefore, we expect that when the input noise ha variance $\sigma^2 < 0.36$, we will obtain better results w error-correcting coding methods. In Sec. V we sh the results obtained for several values of σ^2.

V. Simulation Results

The training set or key vector for our projecti filters consisted of 16 images of letters (capitals *A*- and small letters *a–h*) from the *New York Times* fo These 64×64 pixel images are shown in Fig. 3. T letter occupies ~20×20 pixels of the entire ima We calculated $F = 4$ filters digitally off-line using t algorithm in Sec. II, with each filter being of dimensi 64^2 and a linear combination of all $2^k = 2^F = 2^4 = 16$ k vector test characters. These four filters were used the columns of the $64^2 \times 4$ matrix at P_2 of Fig. 1. T four vector inner product outputs at P_3 of Fig. 4 repi sent the n = 4-bit coded vector **u** (before error-corre tion encoding) that denotes the object class (the inp letter and if it is a capital or lowercase letter). T sixteen coded vectors **u** and the letters to which th correspond are noted in Table III. These also repi sent the actual P_3 output obtained from Fig. 1 (simulation) for the case of $F = 4$ filters and $L =$ output levels.

We then synthesized a second associative process matrix with error correcting using the $(n,k) = (7$ Hamming code. The associative matrix now consist of $n = 7$ filters of 64^2 elements each as the colur vectors in the $64^2 \times 7$ matrix at P_2 of Fig. 1. Each of t $F = n = 7$ filters was again a linear function of sixteen original key vectors and these filters were c

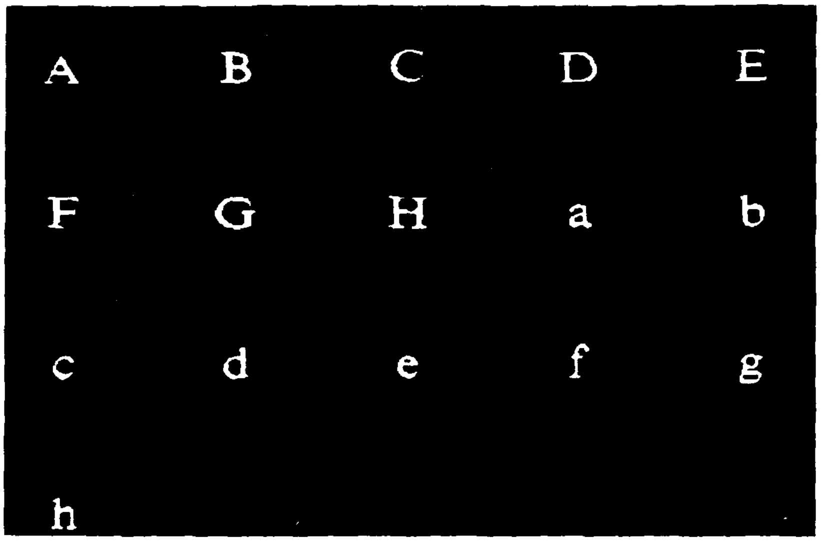


Fig. 3. 64 × 64 pixel training
or key vector. Input images fr
The New York Times font (e
image is 64 × 64 pixels).

Table III. Nonerror-Correcting Output with Training Set (No Noise), $F = 4$, $L = 2$

Letter	Code word	Letter	Code word
A	0000	*a*	1000
B	0001	*b*	1001
C	0010	*c*	1010
D	0011	*d*	1011
E	0100	*e*	1100
F	0101	*f*	1101
G	0110	*g*	1110
H	0111	*h*	1111

Table IV. Reference Key Vector Images and Associated Output C
Words from the (7,4) Hamming Coded Associative Processor with *F*
= 2 and No Noise

Letter	Code word	Letter	Code wor
A	0000000	*a*	1010001
B	1101000	*b*	0111001
C	0110100	*c*	1100101
D	1011100	*d*	0001101
E	1110010	*e*	0100011
F	0011010	*f*	1001011
G	1000110	*g*	0010111
H	0101110	*h*	1111111

culated by a straightforward extension of the method outlined in Sec. II. The $n = 7$ -bit output associated code words (the chosen projection values used in the algorithm) for the sixteen key vectors are given in Table IV. There are also the noiseless P_3 outputs obtained from Fig. 1.

In Table V we present a summary of our results when noise was added to the output vector from the associative processor. We varied σ^2 in order to determine the noise level for which the error-correcting coding scheme is advantageous, and to determine the improvement it provided over a nonerror-correcting code. For each value of σ^2, Table V gives the percent of the sixteen key vector images correctly classified for

Table V. Performance of 4-Bit vs 7-bit Hamming Code Associa
Processors for Various Levels of the Noise Variance σ^2 (the Total N
of Images is 16)

σ^2 Noise	No error correction 4-bit code percent correct (number of errors)	Hamming code percent correct (number of errors)	Number corrected e in Hamming process
0.15	81% (3)	100% (0)	7
0.20	50% (8)	81% (3)	9
0.25	50% (8)	62% (6)	3
0.40	38% (10)	56% (7)	4
0.50	38% (10)	44% (9)	6
0.60	31% (11)	31% (11)	5

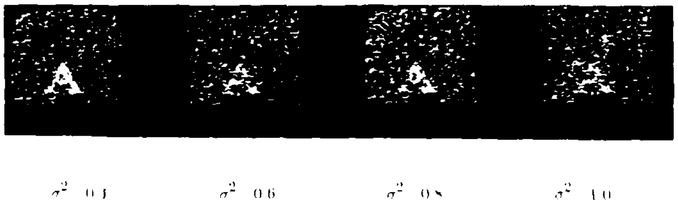


Fig. 4. Noisy *A* with σ^2 varied.

the 4-bit and 7-bit error-correcting Hamming code with the number of errors in parenthesis. Note that any error in one of the 4-bit outputs will be an error and that outputs with two or more bit errors will be errors for the Hamming code associative processor. The last column gives the number of errors (out of a maximum of 16) corrected in the 7-bit error-correcting processor.

From the results in Table V, the error-correcting coding provided better results for values of $\sigma^2 \leq 0.5$. However, the results are significantly better for $\sigma^2 < 0.4$ (classification is 81% for $\sigma^2 = 0.2$ and 100% for $\sigma^2 = 0.15$) and only 6% better than the 4-bit scheme for $\sigma^2 = 0.5$. Thus for large noise levels, the improvement obtained by error-correcting encoding is less significant and not necessarily worth the added memory storage and calculations. Significantly better performance occurs for lower noise levels in agreement with the theory in Sec. IV. In Table VI we list the 4- and 7-bit outputs obtained at P_3 of Fig. 1 for the case of noise with $\sigma^2 = 0.2$. In the output from the 4-bit projection filter associative processor, an * indicates an error. In the output from the 7-bit Hamming code scheme, an * indicates an uncorrectable error, i.e., 2 or more bits in error, and ** indicates a correctable error.

In the previous noise tests, the noise was added directly to the output recollection vector (since for this case we could obtain a theoretical performance estimate). To determine the effect of noise in the input image on the probability of an incorrect bit in the output plane, we require simulations. There is no method to directly calculate this relationship mathematically since each image and noise representation will behave differently. To estimate the amount of input plane noise for which the error-correcting coded output will provide a higher classification rate than the nonerror-correcting coded output, we varied the amount of noise (measured by σ^2) added to the input training images. We could then approximate the probability p of an incorrect bit by the number of incorrect bits in the output divided by the total number of bits. We use ten realizations of the noise for each σ^2 value to obtain better statistics. Zero-mean Gaussian noise (with a specified σ^2) is added to each pixel in the image and the pixel is rethresholded at 0.5 to obtain the noisy binary input image. Figure 4 shows sample versions of the letter A with varying degrees of noise. Notice that the additive zero-mean noise clutters the background as well as drops out data from the letter.

We performed ten runs for each σ^2 value for all sixteen original key image vectors for the 4-bit and 7-bit output coded associative processor. For each σ^2 value, there are sixteen images, with 4 and 7 output bits from the processor and ten runs. The total number of bits considered was (10 runs)*(4 + 7 bits)* (16 images) = 1760. From the number of bit errors out of the total of 1760, we estimate p for the different σ^2 values. The results are shown in column 2 of Table VII. For input noise variance of 0.4–1.0, the estimated value of p ranges from 0.02 to 0.13. Figure 4 shows how poor the input SNR is even with $\sigma^2 = 0.6$. The percentage of

Table VI. Output from $\sigma^2 = 0.2$ Output Noise Tests

Letter	4-Bit code output	(7,4) Hamming code output	Corrected Hamming code
A	0010*	0000001**	0000000
B	0001	0111000*	
C	0010	0110110**	0110100
D	0011	1011001*	
E	1100*	1110010	
F	0001*	0001011*	
G	1011*	1010110**	1000110
H	0111	0101100**	0101110
a	0000*	1010011**	1010001
b	1001	1111001**	0111001
c	1000*	1101101**	1100101
d	1001	0001101	
e	1100	0000011**	0100011
f	1100*	1001011	
g	1110	0010011	
h	1111	0111111**	1111111

Note: For the 4-bit code result the * indicates error; for the 7-b (7,4) Hamming code result the * indicates uncorrectable error; the * indicates correctable error.

Table VII. Estimated Probability p of an Output Bit Transition Error for Input Images with Various Noise σ^2

Noise variance σ^2	Bit error probability p	4-Bit output (number of errors)	Hamming code (number of errors)
0.4	0.02	89% (18)	93% (10)
0.5	0.04	81% (34)	95% (8)
0.6	0.08	73% (44)	91% (15)
0.7	0.10	65% (45)	85% (20)
0.8	0.11	63% (59)	80% (32)
1.0	0.13	54% (73)	69% (50)
1.3	0.18	43% (92)	63% (59)

Note: Each p estimate is based on ten runs. The percentage of th total 160 images for each σ^2 value correctly classified for the 4-b Hamming code schemes are listed (with the number or images mi classified given in parenthesis).

the 160 images (16 characters × 10 runs) per σ^2 valu classified correctly and the number of errors are i cluded in parenthesis for the two coding schemes. F the (7,4) Hamming code, the percentage correctly cla sified includes those output codes which originally ha 1-bit error which were corrected by the postprocessin

In all cases, the error-correcting coding provide considerably improved classification rates and perfo mance compared to the four-filter (or 4-bit) outpu As the input noise variance increases, the classificati rate is lowered for both the error-correcting and no correcting associative processors. For $\sigma^2 = 1.3$, v estimate p at 0.18, which is close to the theoretical lim (of 0.2) estimated in Sec. IV. In this case, the classi cation rate for the Hamming code processor is on 63%; however, it is still a significant improvement ov the 43% classification rate for the four-filter outp As σ^2 is increased further, we find the classificati rate for both the error-correcting and nonerror-c recting processors to be too small to be useful. seen, a considerable amount of input noise can tolerated and the error-correcting associative proce sor will still perform well. In Sec. VI we discuss alternative more advanced codes which are able to correct more bit errors.

VI. Advanced Considerations

From the results presented in the previous section, it is evident that coding schemes can significantly improve classification results in the presence of noise in the input and output. Two more issues we will consider here are (1) the handling of more classes and (2) the correction of more bit errors. We consider binary output vectors initially. In the nonerror-correcting F-filter case, we can increase the number of objects to be recognized by *increasing* $F = k$, *the number of filters* and bits in the output code. In an F-filter scheme with $F = 6$, we can handle up to 2^6 different objects. This would be sufficient to classify the entire alphabet (both lower case and uppercase) along with the ten numeric characters 0–9. If we wanted to extend this to an error-correcting linear block coding scheme, we would need an (n,k) code with k at least equal to 6. If we also wish to implement a coding scheme that is able to correct more than 1-bit error, we will need to synthesize and store more filters (for the extra redundant bits). Since a Hamming code can only correct 1-bit error, we must use other available coding schemes. Binary Boce, Chaudhuri, and Hocquenghem (BCH) codes are one viable alternative.[18,19] For example, there exists a (15,7) BCH code that could handle 128 classes and correct 2-bit errors, but fifteen projection filters must be used. With thirty-one projection filters we could implement a (31,6) code that could handle the alphabet and correct up to 7-bit errors.

BCH codes require complicated decoding techniques. We do not provide all the details, but rather will briefly outline the procedure in order to compare the difficulty. With BCH codes, the syndrome **s** is still calculated by a matrix–vector multiplication such as $\mathbf{r}\mathbf{H}^T$, but **s** is now a $1 \times 2t$ vector (where t is the number of bit errors we desire the code to correct). The elements s_k of **s** will now be the sum of powers of a parameter α, i.e.,

$$\begin{aligned} \mathbf{s}_1 &= \alpha^{j_1} + \alpha^{j_2} + \ldots \alpha^{j_v}, \\ \mathbf{s}_2 &= \alpha^{2j_1} + \alpha^{2j_2} + \ldots \alpha^{2j_v}, \\ &\vdots \\ \mathbf{s}_{2t} &= \alpha^{2tj_1} + \alpha^{2tj_2} + \ldots \alpha^{2tj_v}. \end{aligned} \tag{13}$$

The coset leader demodulated vector for this case has as its elements j_k (k = 1 to v). The values of the j_k indicate the locations of the errors in the original input. To decode these output BCH **s** codes is more difficult but can be realized by an iterative algorithm[14] that solves Eq. (13) for α, v, and then all j_k. Since $v < t$, there are multiple solutions to the set of equations in (13), and the solution that yields an error pattern with the smallest number of errors is the correct solution. Furthermore, other codes exist which can correct a number of errors (t) and can also detect if more than t errors have occurred. With such a code, a vector output can be classified as undecided, and, if desire
input can be reprocessed until no uncorrectable
occur.

All of our previous examples used binary codir
preferable coding scheme would employ multilev
ters with multilevel output coding vectors. W
levels and k bits, the output could handle L^k difl
objects instead of only 2^k as with a binary code.
would significantly enhance the ability of the syst
handle more information with fewer filters. One
multilevel code is a nonbinary version of the
code, the Reed-Solomon code.[20] The decoding fc
more complicated than for the BCH code, but it c
used.

We now consider methods to reduce the size
associative processor matrix (at P_2 of Fig. 1).
optical implementation, the number of filters and
dimensionality are restricted by the size of the s
light modulator on which the matrix is recorded
Using a liquid crystal TV (with 127 × 143 pixels)
we could handle 127 filters, but each can only b
long. If the input key vectors are lexicographic i
plane vectors, the input image size is quite li
($\simeq 10 \times 14$ pixels). By representing the input
feature vector instead of the full image, we can si
cantly reduce the dimensionality, achieve shift-i
ance and some degree of automatic distortion i
ance. The features chosen are dependent on the
of input data and the properties required of the s
(such as shift, rotation, or translation invaria
Typical feature spaces are Hough transforms, Fc
transform coefficients, chord distributions, radia
angular moments, and Fourier-Mellin coefficier
25

The concepts presented here can also be extenc
encoding the outputs of several correlation fil
Correlation filters are implemented and used
differently from the projection filters. A full co
tion of the filters with the input image is perfo
(not just an inner product). The output correl
planes are then searched for peak values above a
fied threshold. These specify the 1 or 0 eler
(peak or no peak) in the output code. The restric
on the number of bits in the code (or the numl
filters) depend on the number of correlations tha
be performed in parallel or rapidly in series (recal
a full correlation must be performed and the
correlation plane searched to obtain 1 bit of the o
code). This is possibly optically.[26]

VII. Summary and Conclusions

We have discussed how to use coding theory t
rect output errors from an optical associative p
sor. The associative processor we use employs p
tion filters for more efficient encoding of inform
Specifically we have demonstrated the ability to
sent 2^k (rather than just k) object classes with a
output recollection vector and a $k \times m$ assoc
matrix, where m is the number of elements in the
key vector. The output code words are select
enable correction of bit transition errors resulting

either output or input noise. We tested the ability of the coding scheme to correct errors for various amounts of noise in the output and input, and we showed that for small bit transition error probabilities ($p < 0.2$), the coding scheme improved results. The example chosen was a sixteen-class binary coding problem using a (7,4) Hamming code with the ability to correct a 1-bit error in the output. Extentions to larger class problems and to increased error-correcting capability were discussed.

We acknowledge the support of this research by General Dynamics, the Defense Advanced Research Projects Agency, and the Air Force Office of Scientific Research. We would also like to thank Ed Baranoski for many fruitful discussions.

References

1. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation," Appl. Opt. **23**, 1620 (1984).
2. W. Chang, D. Casasent, and D. Fetterly, "SDF Control of Correlation Plane Structure for 3-D Object Representation and Recognition," Proc. Soc. Photo-Opt. Instrum. Eng. **507**, 9 (1984).
3. D. Casasent and W.-T. Chang, "Correlation Synthetic Discriminant Functions," Appl. Opt. **25**, 2343 (1986).
4. T. Kohonen, *Self Organization and Associative Memory* (Springer-Verlag, New York, 1984).
5. G. Hinton and J. Anderson, *Parallel Models of Associative Memory* (Laurence Erlbaum Associates, Hillsdale, NJ, 1981).
6. S. Oh, "Walsh-Hadamard Based Distributed Storage Device for the Associative Search of Information," IEEE Trans. Pattern Anal. Machine Intell. **PAMI-6**, 617 (1984).
7. A. D. Fisher, C. L. Giles, and J. N. Lee, "An Adaptive Associative Optical Computing Element," in *Technical Digest, Topical Meeting on Machine Vision* (Optical Society of America, Washington, DC, 1985), paper WB4.
8. N. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical Implementation of the Hopfield Model," IEEE Trans. Inf. Theory **IT-31**, 461 (1985).
9. A. Yariv and S K. Kwong, "Associative Memories Based o Message-Bearing Optical Modes in Phase-Conjugate Resonators," Opt. Lett. **11**, 186 (1986).
10. B. H. Soffer, G. J. Dunning, Y. Owechko, and E. Marom, "Ass ciative Holographic Memory with Feedback Using Phase-Cor jugate Mirrors," Opt. Lett. **11**, 118 (1986).
11. M. Sakaguchi, N. Nishida, and T. Nemotor, "A New Associativ Memory System Utilizing Holography," IEEE Trans. Compu **C-19**, 1174 (1980).
12. B. V. K. V. Kumar and B. L. Montgomery, "Nearest Neighbo Non-Iterative Error-Correcting Optical Associative Processor. Proc. Soc. Photo-Opt. Instrum. Eng. **638**, 83 (1986).
13. D. Casasent and S. Liebowitz, "Model-Based Knowledge-Base Optical Processors," Appl. Opt. **26**, in press (1987).
14. S. Lin and D. Costello, Jr., *Error Control Coding: Fundamental and Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1983)
15. E. R. Berlekamp, *Algebraic Coding Theory* (McGraw-Hill, Ne York, 1968).
16. W. Peterson and E. Weldon, *Error-Correcting Codes* (MI Press, Cambridge, MA, 1973).
17. R. McEliece, *The Theory of Information and Coding* (Addiso Wesley, Reading, MA, 1977).
18. R. C. Boce and D. K. Ray-Chaudhuri, "On a Class of Err Correcting Binary Group Codes," Inf. Control **3**, 68 (1960).
19. A. Hocquenghem, "Codes corecteurs d'erreurs," Chiffres **2**, 14 (1959).
20. I. S. Reed and G. Solomon, "Polynomial Codes over Certai Finite Fields," J. Soc. Ind. Appl. Math. **8**, 300 (1960).
21. G. R. Gindi and A. F. Gmitro, "Optical Feature Extraction vi the Radon Transform," Opt. Eng. **23**, 499 (1984).
22. D. Casasent, "Coherent Optical Pattern Recognition: A Review," Opt. Eng. **24**, 26 (1985).
23. D. Casasent and W-T. Chang, "Generalized Chord Transformation for Distortion-Invariant Optical Pattern Recognition. Appl. Opt. **22**, 2087 (1983).
24. J. Reddi, "Radial and Angular Moment Invariants for Imag *Identification,*" *IEEE Trans.* Pattern Anal. Machine Intel **PAMI-3**, 240 (1981).
25. T. Yatagai, K. Choji, and H. Saito, "Pattern Classification Usin Optical Mellin Transform and Circular Photodiode Array," Op Commun. **38**, 162 (1981).
26. D. Casasent, "Optical AI Symbolic Correlators: Architectur and Filter Considerations," Proc. Soc. Photo-Opt. Instrum. En **625**, 220 (1986).

8. OPTICAL PROCESSOR ASSOCIATIVE MEMORY MAPPING FORMULATIONS

Reprinted from Applied Optics, Vol. 26, page 3641, September 1, 1987
Copyright © 1987 by the Optical Society of America and reprinted by permission of the copyright ow

Optical associative processor for general linear transformations

Raghuram Krishnapuram and David Casasent

A new technique for the realization of general linear transformations using associative memories i
An optical architecture for its implementation is also presented. A low-level feature space proc
this architecture is proposed. The processor is capable of recognizing and locating objects of var
and uses certain linear transformations in the feature space for distortion invariance.

I. Introduction

Linear transformations have been used extensively in the literature to produce feature spaces for pattern recognition. Transforms such as the Fourier transform,[1] Mellin transform,[2] and Hough transform[3] provide feature spaces for pattern recognition. These transformed spaces generally make object detection and identification easier by emphasizing or bringing out certain features of the input image. These transforms can also be made invariant to certain types of distortion of the object. They also achieve a certain amount of dimensionality reduction so that the number of samples required to represent the input image for the purposes of pattern recognition is small. In this paper, we consider the Hough transform for specific detailed realization, although the fundamental mapping, transformation, and associative processor techniques are quite general.

There are many reasons for considering the Hough transform (HT). It is one such feature space which facilitates the detection of a particular shape.[3] It is very attractive because it can be implemented optically in real time and because it is a low-level feature space and is thus quite unique for parallel optical realization. The HT has been defined in a variety of ways.[3-5] It was originally formulated for the detection of straight lines in the input image. It has also been generalized for the detection of other analytical curves (e.g., ellipses,[6] parabolas[7]) and even arbitrary shapes.[8]

All these transformations are linear, and a majority of the HT ones are space-invariant, i.e., the shape of

the curve to which each input point maps is
regardless of the position of the input. These
ties are invaluable, especially for optical imp
tions of these transforms, as will be shown la
also show how the straight-line Hough tra
be used very effectively for pattern recogniti
straight-line Hough space has several advant
can be very easily computed optically, ac
mensionality reduction, and can be made inv
input object distortions by the use of certa
transformations. It can also be used for de
tion of curved objects. Digital methods to
the linear transformations required for the
slow and computationally expensive. Optic
ods to achieve the straight-line HT exist and
be more practical and real-time. In this p
advance an alternative method to compute th
other linear transformations optically using
ciative memory architecture. This approa
tremely general. It can be used to achieve ge
HTs and, in general, any linear transformatio
transformation is also shift-invariant, it can
mented in a very simple and elegant man
acoustooptic cells as we will detail later.
provide a low-level optical associative proce
tem based on the associative memory (AM
ture. The system produces the straight-line
feature space for recognition and location of
arbitrary shapes. This is achieved by the use
transformations, as we will describe.

Section II describes our new associative
approach to a general linear transformatio
algorithm to obtain the required memory mat
tion III discusses an associative processor for
of several linear Hough space transformati
attention to their use for object recognition
shift-invariant property. Section IV advance
new acoustooptic (AO) architectures for th
realization of the proposed associative mem

The authors are with Carnegie Mellon University, Department of Electrical Computer Engineering, Pittsburgh, Pennsylvania 15213

Received 7 February 1986.
0003-6935/87/173641-08$02.00/0
© 1987 Optical Society of America

tion IV describes a proposed low-level optical associative processor system for object recognition. Section VI gives our summary and conclusions.

II. AM Realization of Linear Transformations

Heteroassociative memories map each vector in the input to a corresponding vector in the output, where the input and output vectors need not be of the same dimension. This fact can be used to achieve linear transformations on 1-D or 2-D inputs, where each point in the input is mapped to a corresponding point (or curve, a set of points) in the output.

A. Vector Representation of Mappings

The mappings to be described apply to any data representation (e.g., feature or symbolic space) but are best described for an input image space. Let N be the total number of pixels in an input image. The 2-D input image can be lexicographically represented by a vector with N components, where each component represents a pixel in the input image. The input vector $\mathbf{x}_i$ corresponding to a particular pixel in the input image will have all zeros in it except in the ith position where it will have a 1. (For the time being, we assume that the input image is binary, but this assumption is not required, as we see later.) Similarly, the output associated with each pixel is represented by a vector $\mathbf{y}_i$ of size M, where M is the total number of pixels in the output. Each output vector will have nonzero values only in those positions that correspond to the set of pixels or curve to which the input pixel maps. The size of the output space can be compressed to a variable resolution, and thus $M \leq N$ is possible and generally $M < N$.

B. Construction of the Heteroassociative Memory

Let $\mathbf{X}$ be a matrix with the N input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ as its columns and let $\mathbf{Y}$ be the matrix with the N corresponding output vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ as its columns. We consider the pseudoinverse associative memory[14] matrix M with the N input–output vector pairs as the key and recollection vectors. In this case,

$$\mathbf{Y} = \mathbf{M}\mathbf{X}, \tag{1}$$

where

$$\mathbf{M} = \mathbf{Y}\mathbf{X}^{+}, \tag{2}$$

and $\mathbf{X}^{+}$ is the pseudoinverse of X given by

$$\mathbf{X}^{+} = (\mathbf{X}^{t}\mathbf{X})^{-1}\mathbf{X}^{t}. \tag{3}$$

Without loss of generality, we can order the input vectors so that $\mathbf{X}$ is an $N \times N$ identity matrix. In this case, $\mathbf{X}$ and $\mathbf{X}^{+}$ are identity matrices and, therefore,

$$\mathbf{M} = \mathbf{Y}, \tag{4}$$

i.e., $\mathbf{M}$ is simply the matrix of output vectors

C. Associative Memory Output for a General In
x

The associative memory described above input vector $\mathbf{x}$ to an output vector $\mathbf{y}$ = reference (key) input vectors satisfy the pro

$$x_i(j) = \delta_{ij},$$

where $x_i(j)$ denotes the jth component of th ence vector $\mathbf{x}_i$, and

$$\mathbf{x}_i^{t}\mathbf{x} = \delta_{ij}.$$

The output $\mathbf{y}$ corresponding to a reference in $\mathbf{x}_i$ is

$$\begin{aligned} \mathbf{y} = \mathbf{M}\mathbf{x} &= \mathbf{Y}\mathbf{x} = [\mathbf{y}_1 \ldots \mathbf{y}_N]\mathbf{x}_i \\ &= \mathbf{y}_1 x_i(1) + \ldots + \mathbf{y}_N x_i(N) \\ &= \mathbf{y}_i , \end{aligned}$$

where the last line follows from Eq. (5). output vectors for the N reference vectors a the desired $\mathbf{y}_i$. We note that the maximum reference input–output pairs we are able equal to the dimensionality of the input vect maximum is possible because the input v orthogonal. In general, the number of inp pairs that can be stored in an associative about an order of magnitude smaller than t sionality of the input vectors.[15]

For the case of a general input vector $\mathbf{x}$ co ing to the full lexicographic ordering of an input image, more than one of its compone nonzero, and the components can take al integer values. The output vector $\mathbf{y}$ corres this input vector will be

$$\begin{aligned} \mathbf{y} &= \mathbf{M}\mathbf{x} \\ &= \mathbf{y}_1 x(1) + \ldots + \mathbf{y}_N x(N), \end{aligned}$$

which is a linear (weighted) combination of ence output vectors $\mathbf{y}_i$. This is exactly what for a linear transformation. Therefore, transformation can be achieved through th associative memory approach. We now d further.

D. Memory Matrix for Linear Shift-Invariant Transformations

Equation (4) gives a way to construct th matrix $\mathbf{M}$ for an associative memory that c any linear transformation. Let us assum transformation is shift-invariant as well as the case of 2-D images, this shift-invarian means that the shape of the curve to which pixel maps does not change, and if the posi nonzero input pixel is translated by a certa the positions of the nonzero output pixels a ed by the same amount. Since our input vectors are simply the lexicographically o sions of the 2-D image data, a 2-D transla image is equivalent to a 1-D translation in tors. This holds as long as the shifted point ctor

each
The
s
(5)
refer-

(6)
ector

(7)
s the
actly
er of
re is
This
s are
itput
ry is
men-

ond-
trary
an be
ssible
ing to

(8)
refer-
eeded
linear
posed
s this

emory
rform
at the
r. In
operty
input
of the
nount.
inslat-
output
d ver-
in the
e vec-
ains in

the input field of view and as long as the dimension of the vector equals the dimension of the full input image. [This also holds when multiple objects are present in the 2-D input. We map input points to output curves, and thus objects (viewed as a sum of points) map to the sum of the output curves.] Potential problems can arise near the boundaries of the 2-D input image if the whole output curve does not fit in the 2-D output size specified. This problem can be overcome by slightly modifying the approach presented here. In the interest of presenting the concept, we do not concern ourselves with this case. Since the input and output translations are equal for the shift-invariant case, it follows that the input and output vectors should be equal in length or $M = N$ for the shift-invariant case.

Thus, since our reference input matrix **X** consists of column vectors $\mathbf{x}_i$ which are just translated versions of one another, for the shift-invariant case, the reference output matrix **Y** also contains $\mathbf{y}_i$ that are translated versions of one another. Specifically, $\mathbf{x}_i$ is obtained by vertically shifting $\mathbf{x}_{i-1}$ by one unit and $\mathbf{y}_i$ is obtained by vertically shifting $\mathbf{y}_{i-1}$ by one unit. Therefore, we can write

$$\mathbf{y}_i(j) = \begin{cases} 0 & j = 1, \\ y_{i-1}(j-1) & 2 \le j \le N. \end{cases} \tag{9}$$

It follows from Eq. (9) that for the case of linear shift-invariant transformations, the matrix **M** is lower triangular and Toeplitz.

E. Memory Matrix for Quasishift-Invariant Transformations

In this paper, our specific concern will be with the straight-line HT for reasons explained in Sec. I. Although most of the generalized HTs are shift-invariant, this is not true of the straight-line HT. However, it is shift-invariant for certain translations. We refer to this property as quasishift invariance. In the case of quasishift-invariant transformations, the memory matrix $\mathbf{M} = \mathbf{Y}$ can be partitioned so that $\mathbf{Y} = [\mathbf{Y}_1|\mathbf{Y}_2|\ldots|\mathbf{Y}_{N_p}]$, where the column vectors in each partition $\mathbf{Y}_i$ satisfy Eq. (9). The corresponding input vector elements can be similarly partitioned so that $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_p}]^T$. Thus, for the case of quasishift-invariant transformations, Eq. (8) becomes

$$\mathbf{y} = \mathbf{Y}\mathbf{x} = \mathbf{Y}_1\mathbf{x}_1 + \ldots + \mathbf{Y}_{N_p}\mathbf{x}_{N_p}. \tag{10}$$

It is possible that the $\mathbf{y}_i$ terms satisfying Eq. (9) are not contiguous in the original (lexicographically ordered) memory matrix **M**. In such a case, the columns of M have to be reordered, and the elements of the input vector also have to be reordered accordingly. This means that the input image will now have to be ordered (or scanned) differently to make the matrix **X** equal to the identity matrix. We now illustrate these points with examples of shift-invariant and quasishift-invariant Hough transforms in the following section.

III. Shift-Invariant and Quasishift-Invariant Hough Transformations

We now give examples of shift-invariant and quasishift-invariant Hough transformations and their asso-
ciative processor formulations. We firs
generalized HT for circles because it is a
of a linear shift-invariant transformati
concentrate on the straight-line HT and
space transformations, since these are ou
cern in this paper.

A. Generalized HT for Circles

We first consider the generalized HT
circles of a given radius r. In this case, ea
in the input image is mapped to a circle of
the location of the center of the circle be
tion of the input point. In other words, t
maps to the curve

$$(x' - x)^2 + (y' - y)^2 = r^2$$

in the (x',y') output plane. The accumul
mappings for all input points yields a peak
with coordinates that denote the center
If the input point (x,y) is translated
amount, the output circle is translated
amount (see Fig. 1). Therefore, in th
memory implementation of this transfo
columns of the matrix **M** are shifted ve
another, and each column $\mathbf{y}_i$ of **M** describe
points on the circle of specified radius r.
shift-invariant transformation. To dete
other radii, a new **M** is necessary for each
line HTs allow for an easier search of circ
ent radii as we see in Sec. III.E. Generali
similarly defined for other curves, but stra
realizations (Secs. III.D and III.E) appear
able, especially when distortions or dif
parameters must be searched.

B. Slope-Intercept Straight-Line Hough Tran

As another example, we consider the
tercept(c) parametrization[5] of the straig
In this case, each point (x,y) in the inpu
straight line in the (m,c) space given by

$$y = mx + c, \quad \text{or} \quad c = -xm + y.$$

This defines a straight line with slope
intercept y in the HT output (m,c) space.
mulation of these mappings for all points
the input gives rise to a peak in the output

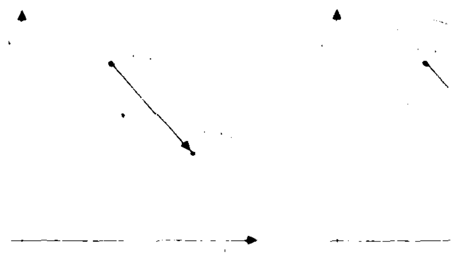


Fig. 1 Example of the shift invariance of the gene
circles (a) input; (b) output of generalize

cuss the
example
We then
e Hough
cific con-

etecting
oint (x,y)
us r with
the loca-
oint (x,y)

(11)

of these
IT space
e circle.
certain
he same
ociative
ion, the
s of one
nonzero
is thus a
ircles of
traight-
f differ-
HTs are
line HT
preferr-
it curve

(m)-in-
ne HT.
ps to a

(12)

and the
e accu-
line in
e (m,c)

HT for

parameters of the input line. If the input point is translated to another location (x',y'), the straight line to which it maps is in general not a simple translation of the straight line in Eq. (12), since both the slope and intercept can change. Thus the mapping is not shift-invariant. If, however, we translate the input point along the y axis only, the slope of the new straight line remains the same, only the intercept changes, and the mapping is a simple translation (in the intercept c by an amount $y - y'$) of the old line given in Eq. (12). We use this to produce a quasishift-invariant transformation. We scan the image vertically and note that all the $\mathbf{y}_i$ terms corresponding to pixels along any vertical line in the input are shifted versions of one another and thus fall into one partition in Eq. (10). Different partitions are required for each column of the input. Thus, the number of partitions is equal to the number of columns in this case. However, when the input is scanned along vertical lines, the $\mathbf{y}_i$ terms that satisfy Eq. (9) are contiguous, and the mapping is easily achieved and used in a quasishift-invariant processor. In all our transformation cases, the partitioning of $\mathbf{X}$ and $\mathbf{Y}$ is such that the number of partitions N_p in Eq. (10) equals either the number of rows or the number of columns in the image. This may not be the case for other transformations.

C. Normal Parametrization of Straight-Line HT

The effect of input shifts on the $\mathbf{y}_i$ vectors in the normal parametrization of the HT is considered next. In this case, each point (x,y) in the input maps to a sinusoid in a (θ,p) Hough space given by

$$p = x \cos \theta + y \sin \theta$$
$$= (x^2 + y^2)^{1/2} \cos [\theta - \tan^{-1}(y/x)]. \tag{13}$$

Equation (13) describes all straight lines that could pass through point (x,y) in terms of their normal distance p from the origin and the angle θ this normal makes with the positive x axis. The accumulation of these mappings for all the points on a straight line in the input produces a peak in the output HT space at the (θ,p) parameters of the line. In general, if the input point is translated to a new position, both the amplitude and phase of the sinusoid to which it maps change. Hence the output mapping is not a simple translation. However, if the input point is translated so that the new point and original point lie on the same circle centered at the origin, the output
translation of the sinusoid given by Eq.
occurs because the sinusoid's amplitude $(x^2$
the new point remains the same, and onl
shifts, as shown in Fig. 2. With this insight,
if the input image is scanned in a polar f
normal HT can be made to be a quasishif
mapping that is shift-invariant for shifts
To avoid scanning the image in a polar f
could perform a simple rectangular-to-polar
dinate transformation of the input image
conventional raster scan. The polar tran
verts the circular translation required for
invariance into a linear translation in ϕ. Th
data are shift-invariant in ϕ but not in r. T
input image is converted to a polar (r,ϕ) r
tion, a normal HT of this polar data will be
invariant and will have partitions of $\mathbf{M}$ wit
that are shifted versions of one another. T
since the transformed input points along ar
allel to the ϕ axis (i.e., points in the origina
any circle centered at the origin) will have
outputs in the Hough space that are trans
sions of one another. Each partition corres
row in the (r,ϕ) representation. Unfortu
though the retangular-to-polar transforma
ear, it is not shift-invariant. Thus the
memory shift-invariant mapping techniqu
cuss cannot be used to implement the polar
One could implement it by computer-gener
gram methods or by a camera with a spec
Therefore, although we can theoretically im
polar coordinate transform and a normal st
HT using an AM architecture, we cannot us
ple and elegant architecture presented in t
The normal straight-line HT is nevertheless
ful for distortion-invariant pattern recogni
plained in the next section and can be easily
using other methods.[9,10] The preferable sys
HT for distortion invariance would thus
techniques to produce the HT and would u
approach to do the other transformations in
space that are required for distortion-invari
location.

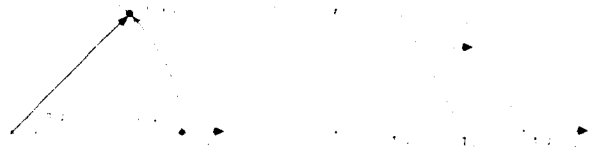


Fig. 2. Example of the quasishift invariance of the straight line HT: (a) input; (b) straight-line HT.

D. Hough Space Transformations for Distortion

We now discuss some of the transformati
straight-line Hough space that can be used
distortion (scale, rotation, and translatio
ant.[12] We consider the normal straight-line
the transformations here are easily desc
made. The normal straight-line HT, as de
Eq. (13), is not invariant to scale, rotation, an
tion changes of the input object. However,
ble to perform transformations in the Houg
that the effects due to these distortions are el

Similar transformations to those discusse
be derived for the slope-intercept straight
but these are much more complicated and
implemented in a simple way. Generalized

also be made distortion invariant but only for one type of object or curve. These restrictions do not apply to the normal straight-line HT, as we describe in what follows.

Let the input object consist of a set of line segments and let (θ,p) be a point in the Hough space corresponding to a line segment in the input object centered at the origin. If the input object is scaled by a scaling factor s, it can be shown[12] that the line segment would map to a new point (θ',p') given by

$$p' = sp \quad \text{and} \quad \theta' = \theta. \tag{14}$$

Equation (14) defines a transformation that maps a point (θ,p) in the Hough space to a point $(\theta',p') = (\theta,sp)$ in the transformed space. Equation (14) notes that the transformation is the same (and hence shift-invariant) for each θ, but it is different (and hence not shift-invariant) for each p. In other words, the transformation is shift-invariant for translations along the θ axis. However, it is not shift-invariant for translations along the p axis. Therefore, this transformation is quasi-shift-invariant.

Similarly, if (θ,p) is a point in the Hough space corresponding to a line segment in the input object centered at the origin and if the input object is rotated by an angle ϕ, it can be shown[12] that the line segment would now map to a different point (θ',p') in Hough space given by

$$p' = p \text{ and } \theta' = \theta + \phi. \tag{15}$$

Equation (15) represents a transformation that can be performed in Hough space to search for different input object rotations. It represents a shift in the Hough space along the θ axis. *Since the shift is independent* of the position of the point, it is a shift-invariant transformation.

Finally, it can also be shown[12] that if the input object is translated by (x_0,y_0), the point (θ,p) will now map to the point (θ',p') given by

$$\begin{aligned} p' &= -p - t\cos(\theta - \alpha) \text{ and } \theta' = \theta + \pi && \text{if} \quad p + t\cos(\theta - \alpha) < 0, \\ p' &= p + t\cos(\theta - \alpha) \text{ and } \theta' = \theta && \text{if} \quad p + t\cos(\theta - \alpha) \geq 0, \end{aligned} \tag{16}$$

where

$$t = (x_0^2 + y_0^2)^{1/2}; \quad \alpha = \tan^{-1}(y_0/x_0). \tag{17}$$

Equation (16) represents a shift along the p axis. The shift is not uniform for all points, but it is the same for all points that have the same θ value. Thus it is a quasi-shift-invariant transformation.

The above transformations for rotation and shift both require the Hough space to be scanned in the direction of the p axis and are shift invariant in p. Thus they can be combined into one quasi-shift-invariant transformation. By performing these transformations for various values of the distortion parameters and comparing (matching) the resultant transformed HT patterns with the HT patterns of various reference objects, we can identify the object in the face of in-plane distortions and also determine its distortion parameters. The associative memory architectures as detailed in Sec. IV can perform thes mations very efficiently and fast. (We changes in scale as changes in the curve p and search them by varying the curve de One measure of how well two HT patterns m point-by-point product of the two HTs. Th is also the correlation value of the two HT p the origin. Thus the matching can be d optical correlation architecture. For the ca 1-D shift search of the HT of the input mu pared vs several reference HT patterns, a r nel AO architecture is possible. Such a 1-D case, as we have shown. If the correlation such comparisons exceeds a predetermined the object is identified. However, comparir terns for all possible distortions and class object is not a trivial task (even with the parellelism of optics). Fortunately, this pr be overcome by treating the input object as arbitrary shape and using the procedure de the next section.

E. Transformations for Detecting Curved Objec

The normal straight-line HT space can al for curve detection. In this case, we first description of the curve in terms of the norm eters p and θ. Let this description be

$$p = T(\alpha_1,\ldots,\alpha_n,\theta),$$

where $\alpha_1,\ldots,\alpha_n$ are the parameters of the cu description is a set of peaks in a 2-D norma line HT of the input curve after thresholding the points below a threshold to zero and k grey-level values of points above the thresl detect a curve and its parameters in an input first form the normal straight-line HT of pattern and threshold it. We then perforr shift-invariant linear transformation of th space given by

$$p' = p - T(\alpha_1,\ldots,\alpha_n,\theta - \phi)$$

and then an inverse *Hough transform*.[13] Th $\alpha_1, \ldots,$ and α_n used in Eq. (19) are the para the curve being searched for and ϕ its rotat The presence of a peak in the inverse HT spa fies the object. The parameters used in the mation in Eq. (19) (that yield a peak in t Hough space) identify the parameters of th curve. Scale changes are viewed as chang values of the curve's parameters α_n. The p the peak in the inverse HT space defines t center,[13] i.e., its shift (x_0,y_0). Thus this allows us to identify a curve's shape, its pa and its shift and rotation. Use of this techni detection of missile trajectories has been elsewhere.[17]

F. Inverse Hough Transform

As a final example of a quasi-shift-invari formation, we consider the inverse HT no For a normal straight-line HT, the inverse

isfor-
view
eters
tion.)
is the
asure
'ns at
iy an
hen a
com-
chan-
is the
from
hold,
pat-
g the
l and
n can
rve of
ied in

used
n the
iram-

(18)

This
ight-
etting
ig the
. To
ge, we
input
quasi-
Hough

(19)

ues of
ers of
value.
lenti-
isfor-
iverse
ecific
n the
ion of
irve's
edure
eters,
in the
ussed

trans-
ibove.
maps

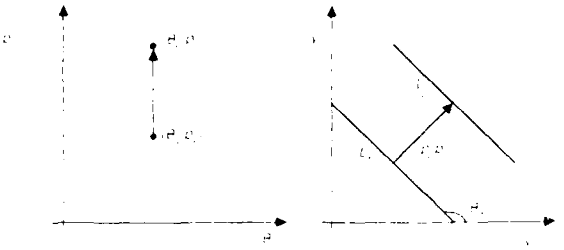


Fig. 3. Example of the quasishift invariance of the inverse HT: (a) HT space; (b) inverse HT space.

each point (θ,p) in the Hough space to a straight line in the (x,y) space (see Fig. 3). It is obvious that if the input pixel at (θ,p) is translated along the p axis, the straight line to which it maps is merely translated in the direction of its perpendicular, and its slope does not change (see Fig. 3). Thus this transformation is shift-invariant in the direction of the p axis. If this transformation is implemented using an associative memory, it will be quasishift-invariant if we scan the input HT along the p axis, and the $\mathbf{y}_i$ terms that are shifted versions of one another will be contiguous.

Section IV describes how the shift-invariant and quasi-shift-invariant transformations can be achieved optically using acoustooptic cells. We use the HT transformations described in this section for specific case studies. Section V advances an associative processor system that is capable of curved object identification and location. The system uses the straight-line HT and the Hough space transformations described in this section.

IV. Optical Realization of the Associative Processor

In this section, we show how a low-level processor based on (quasi)shift-invariant linear transformations can be optically realized using acoustooptic (AO) cells. It is a low-level processor, in the sense that it operates on raw image data extracting local low-level iconic image features (e.g., lines, edges, and their slopes) and preserves most of the input data information. We first describe an architecture that can perform general linear shift-invariant transformations. We then describe a different architecture, which is capable of performing general quasishift-invariant transformations. We would like to restate that these architectures are capable of realizing any general linear shift-invariant and quasishift-invariant transformations, but we focus our attention on the normal straight-line HT, because it can be used to recognize objects of arbitrary shapes, and it can be made distortion-invariant. As noted in the previous section, the transformations required to achieve this are quasishift-invariant and can be easily achieved using the architecture suggested in this section. However, we recommend obtaining the normal straight-line HT using the rotating prism method[10] because we need to sample in input in a polar fashion if we want to use the associative processor architecture suggested in this section to ge

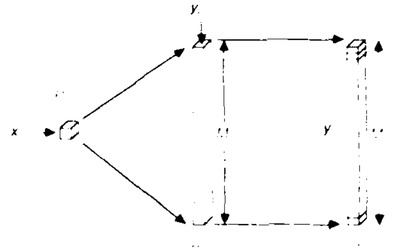


Fig. 4. Optical realization of shift invariant transforma

the HT.

We see from Eq. (8) that the output of an linear transformation is given by the sum of the ence output vectors weighted (multiplied) by th responding elements of the input vector. If the ence output vectors $\mathbf{y}_i$ are shifted versions another, we can acheive this linear transformat the simple optical matrix-vector processor sho Fig. 4. This architecture consists of a point mod at plane P_1, the output of which is expanded to i nate uniformly an AO cell at plane P_2. The leaving the AO cell is then imaged onto a 1-D de array at plane P_3 which integrates in time. assume that the AO cell can be divided into M lengthwise, where M is the number of elements The vector $\mathbf{y}_1$ is first fed to the AO cell, and th ments of the input vector $\mathbf{x}$ are fed to a point mo tor at P_1. As $\mathbf{y}_1$ propagates downward in the AO automatically creates $\mathbf{y}_2$, $\mathbf{y}_3$, etc. as these are s versions of $\mathbf{y}_1$. Thus, by pulsing P_1 with the ele of $\mathbf{x}$ at intervals of T_A/M (where T_A is the apertur length of the AO cell) and time-integrating c detectors at P_3 over N intervals each T_A/M, we a the weighted sum of the $\mathbf{y}_i$ as required by Eq. (8). the case of linear shift-invariant transformation M, as noted in Sec. II. D.) Since we have to load the AO cell before we can start the computation total time T required to obtain the output is

$$T = T_A + NT_A/M = (1 + N/M)T_A.$$

In practice, the AO cell cannot be divided into M [M is the time–bandwidth product (TBWP) of th cell], because M is rather large for most cases. example, consider the case of a generalized H circles for a 128 × 128 image. We have N = 16,000. If the AO cell can only accommodate a T of m (where $m < M$), we operate the processo obtain m of the M output elements at the end of m s. We then shift out the contents of the det and repeat the process M/m times to obtain the output. From Eq. (20), the total time T_1 tak produce the output on an m element processor i

$$T_1 = (M/m)(1 + N/m)T_A$$

For $N = M = 128 \times 128$, $m = 500$, and $T_A = 5\ \mu$s, w

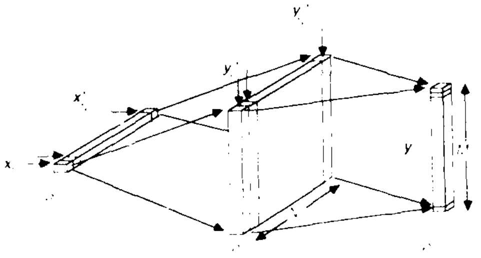


Fig. 5. Optical realization of quasishift-invariant transformations.

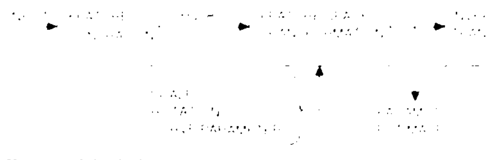


Fig. 6. Block diagram of the proposed optical associati
system.

$T \approx 5$ ms in Eq. (21), and the point modulator at P_1 has to be pulsed with the elements of $\mathbf{x}$ at a rate m/T_A = 100 MHz. This is a very realistic data rate for the point modulator and AO cell.

The above architecture realizes a shift-invariant linear transformation. However, if we are using the normal straight-line HT for object recognition, many of the transformations that we need to perform in the Hough space are quasi-shift-invariant. We now consider the use of multichannel AO cells to achieve quasi-shift-invariant transformations. The architecture we consider is shown in Fig. 5. Similar architectures have been suggested for high-accuracy vector inner product processors.[18] The input plane P_1 consists of a row of N_c point modulators where N_c is the number of channels in the AO cell. The multichannel AO cell is placed at P_2. Each channel consists of m regions (TBWP = m) as in the previous architecture. The light from each point modulator is expanded to illuminate a corresponding AO cell channel, and the light leaving the different channels is summed and imaged onto a 1-D detector array as shown. Let the number of partitions in the memory matrix $\mathbf{Y}$ be N_p, as discussed in Sec. II. E, where the $\mathbf{y}_i$ terms in each partition are shifted versions of one another. Let us assume that we have an AO cell with $N_c = N_p$. We feed one $\mathbf{y}_i$ (the first $\mathbf{y}_i$) of each partition to one of the N_c different AO channels. Each AO channel is assumed to have TBWP of m. The input vector $\mathbf{x}$ is also partitioned (and rearranged in some cases) so that $\mathbf{x} = [\mathbf{x}'_1,\mathbf{x}'_2,\ldots,\mathbf{x}_{N_p}]^T$ as detailed in Sec. II.E. These $\mathbf{x}_i$ terms are time-sequentially fed to the corresponding N_c point modulators. The system in Fig. 5 can be thought of as an N_c channel version of the one in Fig. 4, with the N_c outputs summed into a common detector array. The different $\mathbf{y}_i$ terms in different channels produce the different terms in Eq. (10), as they propagate through the different channels. Thus the whole matrix–vector product is achieved at the end of $(T_A/m)n$ s, where n is the maximum number of $\mathbf{y}_i$ terms in any partition (i.e., the maximum number of shifted versions of the $\mathbf{y}_i$ needed in any partition). As in Eq. (21), we repeat this (M/m) times for M element outputs greater than the TBWP = m of the AO cell. The number of partitions can be greater than the number of channels. In this case, we repeat the above procedure N_p/N_c times to achieve the complete matrix–vector product in Eq. (10). Therefore, the total time T_2 required to complete the transformation is gi

$$T_2 = (N_p/N_c)(M/m)(1 + n/m)T_A.$$

If the number of $\mathbf{y}_i$ terms in each partition is t
$n = N/N_p$. As an example, we consider u
processor to compute the inverse HT. We
the case when $N = 72 \times 25$ (the size of the HT s
$= 128 \times 128$ (the size of the image or inverse H
$N_p = 72$ (number of partitions, one for each θ v
$= 36$ (number of channels in the AO cell),
(TBWP of each AO cell), and $T_A = 5$ μs. For
Eq. (22) gives $T_2 \approx 330$ μs. Therefore, the p
architecture is quite fast and realistic.

V. Proposed Low-Level Optical Associative Pr

Figure 6 shows the block diagram of the p
low-level optical associative processor. The
line HT of the input image is first computed.
be achieved in a variety of ways, including
method presented in this paper. It is, howeve
able to use the rotating prism method.[10] (Th
to be the most practical technique, since the A
od requires that we scan the input image ir
fashion or perform a rectangular–polar tra
tion.) The HT obtained is operated on by an
tive processor (performing quasishift-invaria
formations) to determine the curve parame
the rotation value for the curve. The opera
quired on the HT are linear and (quasi)shift-i
as explained in Sec. III. Hence the architec
gested in Fig. 5 can be used to perform thes
tions. The same architecture is then used to
the inverse HT to provide the translation pa
of the object. Thus this processor can be reali
one HT unit and two AO cell AM units of
shown in Fig. 5.

Some advantages of using this technique a
below. We use the normal straight-line HT
objects of all shapes and thus avoid the use of
generalized transforms for objects of differen
The transformed spaces are always 2-D, whi
simpler and more efficient use of memory. T
od also works for multiple objects and partia
Other associative memory techniques first u
toassociative memory to map partial object
objects and then a heteroassociative memory
object identification. Since our method work
tial objects, we do not need the autoassociati

ry. The use of different transformations in the Hough space and an inverse transform to achieve the object identification and location in our associative memory mapping is more efficient than a more conventional autoassociative memory followed by a conventional heteroassociative memory that maps every possible distorted version of the various objects to appropriate output vectors.

Summary and Conclusions

In this paper, a new approach to achieving linear transformations on 2-D images using associative memories has been suggested. We have shown how general linear transformations can be viewed as associative memories. We have detailed the different linear transformation operations required for the case of an HT feature space for pattern recognition and how each can be achieved by an AM processor. These include the Hough transform, the HT space transformations, and an inverse Hough transform. The construction of the memory matrix required for each associative memory processor has been detailed, and an architecture for its optical realization has been suggested. The architecture is simple, elegant, and capable of real-time processing for shift-invariant as well as quasi-shift-invariant linear transformations. We have thus suggested a low-level associative processor that uses linear transformations for the recognition and location of curved objects. The processor can be implemented optically.

The support of this research by a grant from the Air Force Office of Scientific Research (grant AFOSR-84-0293) as well as partial support from the Office of Naval Research is gratefully acknowledged. Fruitful discussions with Bradley Taylor are also acknowledged.

References

1. D. Casasent and V. Sharma, "*Fourier Transform Featur* Studies," Proc. Soc. Photo-Opt. Instrum. Eng. **149,** (198
2. D. Casasent and D. Psaltis, "New Optical Transforms f tern Recognition," Proc. IEEE **65,** 77 (1977).
3. P. V. C. Hough, "Method and Means for Recognizing C Patterns," U.S. Patent 3,069,654 (1962).
4. R. O. Duda and P. E. Hart, "Use of the Hough Transf Detect Lines and Curves in Pictures," J. Assoc. Comput **15,** 11 (1972).
5. D. H. Ballard and C. M. Brown, *Computer Vision* (Pr Hall, Englewood Cliffs, NJ, 1982).
6. S. Tsuji and F. Matsumoto, "Detection of Ellipses by a M Hough Transformation," IEEE Trans. Comput. **Com-** (1978).
7. H. Wechsler and J. Sklansky, "Finding the Rib Cage ir Radiographs," Pattern Recognition, **9,** 21 (1977).
8. D. H. Ballard, "Generalizing the Hough Transform to Arbitrary Shapes," Pattern Recognition, **13,** 111 (1981).
9. *G. Eichman and B. Z. Dong, "Coherent Optical Produc* the Hough Transform,' Appl. Opt. **22,** 830 (1983).
10. G. R. Gindi and A. F. Gmitro, "Optical Feature Extrac the Randon Transform," Opt. Eng. **23,** 499 (1984).
11. P. Ambs, S. H. Lee, Q. Tian, and Y. Fainman, "Optical mentation of the Hough Transform by a Matrix of Holog Appl. Opt. **25,** 4039 (1986).
12. R. Krishnapuram and D. Casasent, "Hough Space Trans tions for Discrimination and Distortion Estimation," C Vision Graphics Image Process. **38,** 299 (1987).
13. D. Casasent and R. Krishnapuram, "Curved Object Loca Hough Transformations and Inversions," Pattern Reco **20,** No 2, 181 (1987).
14. T. Kohonen, *Self Organization and Associative M* (Springer-Verlag, New York, 1984).
15. G. S. Stiles and D. Denq, "On the Effect of Noise on the *Penrose Generalized Inverse Associative Memory,'* Trans. Pattern Anal. Machine Intell. **PAMI-7,** 358 (198
16. D. Casasent and M. Kraus, "A Polar Camera for Space- Pattern Recognition," Appl. Opt. **17,** 1559 (1978).
17. D. Casasent and R. Krishnapuram, "Detection of Target tories Using the Hough *Transform*," Appl. Opt. **26,** 247
18. D. Casasent and B. K. Taylor, "Banded-Matrix High mance Algorithm and Architecture," Appl. Opt. **21,** 1476

9. STORAGE CAPACITY AND DECISION MAKING ASPECTS OF OPTICAL ASSOCIATIVE PROCESSORS

Vol. 848
November 1987
818 51

Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results

David Casasent and Brian Telfer
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

The storage capacity, noise performance, and synthesis of associative memories for image analysis are considered. Associative memory synthesis is shown to be very similar to that of linear discriminant functions used in pattern recognition. These lead to new associative memories and new associative memory synthesis and recollection vector encodings. Heteroassociative memories are emphasized in this paper, rather than autoassociative memories, since heteroassociative memories provide scene analysis decisions, rather than merely enhanced output images. The analysis of *heteroassociative memories has been given little attention.* Heteroassociative memory performance and storage capacity are shown to be quite different from those of autoassociative memories, with much more dependence on the recollection vectors used and less dependence on M/N. This allows several different and preferable synthesis techniques to be considered for associative memories. These new associative memory synthesis techniques and new techniques to update associative memories are included. We also introduce a new SNR performance measure that is preferable to conventional noise standard deviation ratios.

1. INTRODUCTION

Much has been written about associative memory storage capacity and the recollection and error correction properties of such memories. Section 2 reviews associative memory synthesis, several of the neural and other associative memory models suggested, and advances initial remarks on the storage capacity of associative memories. The similarity of associative memory matrix rows to pattern recognition linear discriminant functions (LDFs) is included. The assumptions on the key vectors in the different associative memories are also noted, since this is not generally given proper attention. As we shall see, most work has considered autoassociative memories (AAMs). In Section 3, we derive expressions to show that heteroassociative memory (HAM) performance and storage capacity are quite different from those of AAMs. We also advance new and preferable performance measures to be employed in comparing such memories. Quantitative supporting data on HAM and AAM comparative noise performance and storage capacity are then advanced in Section 4. We conclude (Section 5) with initial remarks on different associative memory synthesis techniques to provide updating and altering of associative memories. Our work and attention to HAMs is especially important in image analysis, image understanding and pattern recognition, rather than image reconstruction and image enhancement as is generally the AAM case considered.

2. SYNTHESIS AND STORAGE

2.1 TERMINOLOGY AND PSEUDOINVERSE ASSOCIATIVE MEMORIES

In our notation, the input key vectors $\underline{x}_k$ are of dimension N, the output recollection vectors $\underline{y}_k$ are of dimension K, there are M key/recollection vector pairs and the associative memory matrix $\underline{M}$ is K × N. An associative memory is intended to output a recollection vector $\underline{y}_k$ that is closest to or most closely associated with a given input key vector $\underline{x}_k$, i.e. we desire $\underline{M}\,\underline{x}_k = \underline{y}_k$ for all $k = 1$ to M. If we form the key vector matrix $\underline{X}$ of size N × M (with the $\underline{x}_k$ as its columns) and the recollection vector matrix $\underline{Y}$ of size K × M (with the $\underline{y}_k$ vectors as its columns), the associative memory must satisfy $\underline{M}\,\underline{X} = \underline{Y}$. If $\underline{X}$ is square and non-singular, the solution to this can be written as

$$\underline{M} = \underline{Y}\,\underline{X}^{-1}. \tag{1}$$

Generally $\underline{X}$ is not square and this solution is not of practical use. The typical solution used is

$$\underline{M} = \underline{Y}\,\underline{X}^{+}, \tag{2}$$

where the pseudoinverse of $\underline{X}$ is

$$\underline{X}^{+} = (\underline{X}^{T}\underline{X})^{-1}\underline{X}^{T} \tag{3}$$

and where the vector inner product (VIP) matrix is

$$\underline{V} = \underline{X}^{T}\underline{X}. \tag{4}$$

The data matrix is denoted as $\underline{X}^{T}$ (it has the $\underline{x}_k$ key vectors as its row vectors). We note that when the $\underline{x}_k$ vectors are orthonormal, then $\underline{V}^{-1} = \underline{I}$ and $\underline{X}^{+} = \underline{X}^{T}$. The solution in Eq.(2), with $\underline{X}^{+}$ given by Eq.(3), is useful since $\underline{X}^{T}\underline{X}$ is a square matrix and hence it has an inverse (if the $\underline{x}_k$ are linearly independent, in which case $\underline{V}$ is of full rank). Thus, this solution in Eq.(3) is only possible when the $\underline{x}_k$ are linearly independent. In other cases, $\underline{X}^{+}$ must be calculated using singular value decomposition and other advanced techniques, which first produce a set of orthogonal vectors, or which form separate linear discriminant functions (each of which is a row of the associative memory matrix). The pseudoinverse solution is an exact solution if the $\underline{x}_k$ are linearly independent (and in this case the simple $\underline{X}^{+}$ solution noted in Eq.(3) can be used). This pseudoinverse solution in Eq.(2) is the minimum mean square error (MSE) solution that minimizes $\|\underline{Y}-\underline{MX}\|^2$. In cases when Eq.(3) can be used, $\|\underline{Y}-\underline{MX}\|^2 = 0$. If the $\underline{x}_k$ are orthonormal, then $\underline{X}^{+} = \underline{X}^{T}$ and calculation of the memory matrix $\underline{M}$ is trivial. When $M < N$, there are more unknowns than equations, and an infinite number of solutions exist (the underdetermined problem) and Eq.(2) is one of these solutions. This pseudoinverse solution is the minimum norm solution [17] to $\underline{M}\,\underline{X} = \underline{Y}$, i.e. it is the solution whose outputs $\underline{y}_k$ are the least effected by input perturbations.

The associative memory described above is a HAM. The typical associative memory discussed is the AAM. In this memory, the prior discussion is still valid with $\underline{Y} = \underline{X}$ and $\underline{M} = \underline{X}\,\underline{X}^{+}$ (thus, the AAM is a special case of the HAM). We feel that more attention should and must be given to HAMs. Kohonen [1] discusses $\underline{X}\,\underline{X}^{+}$ as the orthogonal projection operator, where the output vector $\underline{y}$ produced is a linear combination of the key vectors with minimum MSE for the case of an AAM.

The AAMs and HAMs described above are the most common associative memories discussed [1]. The use of the data matrix $\underline{X}^{T}$ as an associative memory has also been suggested and shown to be a preferable nearest neighbor associative memory for binary [2] and gray scale key vectors. The technique by which the associative memory is formed can be used to distinguish different associative memory systems. In one model [4,5,6], the memory is formed from data matrices of the key and recollection vectors in a VIP processor. The most common synthesis technique discussed forms the matrix as the sum of the vector outer products of key and recollection vector pairs [1]. Some specific associative memories [8] restrict the key vector elements to be 0 or ± 1. In synthesis, they sum the vector outer product (VOP) of each vector pair and quantize the final matrix to 0 or ± 1. In other cases, the diagonal elements of the memory matrix are set to 0 (usually to model neural networks). In some memories, recollection occurs after one matrix-vector multiplication. In other cases, the output from each matrix-vector multiplication is thresholded and fed back to the input of the system, and the final recollection output is obtained only after several iterations. In one of the most popular associative memories, the Hopfield memory [9,10], the key and recollection vectors are bipolar binary and the diagonal elements of the matrix are 0. Some associative memories require sparse key vectors for efficient recall. Most associative memories are synthesized as matrix-vector processors. However, analogous holographic associative memory synthesis techniques also exist [11,12,13].

Thus, there are a large variety of associative memories. We consider HAMs and gray-level *memories and key vectors. Our general preference in image analysis is to use* $\underline{x}_k$ *input key vectors* that have no unrealistic constraints (such as linear independence, orthogonality, etc.). In a subsequent paper, we detail techniques to achieve this and provide examples of ways to achieve the more important property of shift invariance in associative processors intended for image processing.

2.2 KEY VECTOR REQUIREMENTS

Generally, key vector image inputs cannot be assumed to be linearly independent, and thus the practical use of associative memories for such image data is of concern. In some cases, linear independence may occur, of course, but this cannot be guaranteed. If the $\underline{x}_k$ are image domain vectors (i.e. lexicographically ordered images), and if $M \ll N$, then often we will find that the $\underline{x}_k$ are independent, or at least there is a reasonable assurance that this will occur. However, we note that there is no guarantee of this. If the $\underline{x}_k$ are feature vectors, then generally $M > N$ and the key vectors are linearly dependent. For the more practical and general case of linearly dependent key vectors, one can employ singular value decomposition [14]. This algorithm produces orthogonal vectors and for the case of linearly independent key vectors it addresses practical numerical stability issues associated with calculations of the inverse of $\underline{V}$. This merits attention, since the condition number of $\underline{V}$ is the square of that of the matrix $\underline{X}$. The problem with the SVD technique is its high numerical computational load, which precludes its use in real time and its use for updating associative memory matrices A modified Karhunen-Loeve approximation to $\underline{X}^{+}$ developed for image domain synthetic discriminant functions is quite useful here also [15] It allows operation on high-dimensionality linearly dependent key vectors. The technique used is to calculate the eigenvectors of the correlation matrix from the much smaller dimensionality VIP matrix. We do this for the key vectors for each class We retain only several (typically 3) eigenvectors per class We then orthogonalize the eigenvectors from all classes (using Gram-Schmidt (GS) or related techniques) All of these calculations are performed in the reduced VIP space, hence allowing real time calculations The memory can then be easily described in terms of the original higher dimensionality image space. These final eigenvectors are then used as the rows of the associative memory matrix We refer to this as the VIP-GS associative memory synthesis technique [3].

The direct synthesis of an associative memory as the sum of vector outer products of each key/recollection vector pair requires orthonormal key vectors (and will not yield correct results even for linear independent key vectors, since $\Sigma \underline{x}_k \underline{x}_k^T = \underline{X}\,\underline{X}^T = \underline{X}(\underline{X}^T\underline{X})^{-1}\underline{X}^T = \underline{X}\,\underline{X}^{+}$ only for orthonormal vectors). Similarly, the simple VIP synthesis of an associative memory also requires orthonromal key vectors. However, when a nonlinearity is used at the intermediate plane [4], where the product of the input vector and the data matrix is formed, the requirement of orthonormal key vectors can be reduced However, if the key vectors are only restricted to be linearly independent, this method will still <u>not</u> achieve proper results The VIP-GS synthesis technique and the iterative Widrow-Hoff are two very attractive and real time techniques for associative memory synthesis in the practical case of linearly dependent key vectors

2.3 ANALOGY WITH PATTERN RECOGNITION LINEAR DISCRIMINANT FUNCTIONS (LDFs)

We now discuss how the different solutions to $\underline{M}\,\underline{X} = \underline{Y}$ are related to different pattern recognition LDFs For linearly independent key vectors, the pseudoinverse solution is related to various synthetic discriminant functions (SDFs) [15] for distortion-invariant pattern recognition, i e the outputs from the pattern recognition system are analogous to the recollection vectors $\underline{y}_k$ in associative memories and the key vector input images $\underline{x}_k$ are analogous to the images to be classified independent of distortions, etc To see this, we consider the filter function (or associative memory vector) $\underline{h}$ to be a linear combination of several key vectors, i e

$$\underline{h} = \Sigma a_j \underline{x}_j = \underline{X}\,\underline{a}, \tag{5}$$

where $\underline{X}$ has the training images or key vectors $\underline{x}_j$ as its columns and the vector $\underline{a}$ has as its elements the coefficients a_j that describe the filter function $\underline{h}$ This filter is the solution $\underline{a} = \underline{V}^{-1}\underline{u}$ to $\underline{V}\,\underline{a} = \underline{u}$ where $\underline{V} = \underline{X}^T\underline{X}$ is the VIP matrix and $\underline{u}$ is the vector of desired outputs whose bit code denotes the class of the input key vector $\underline{x}$ under test The filter function, when written as a row vector is thus the following solution

$$\underline{h}^T = \underline{u}^T(\underline{X}^T\underline{X})^{-1}\underline{X}^T = \underline{u}^T\underline{X}^{+}. \tag{6}$$

This solution is the same as Eq.(2), where each row in the pseudoinverse memory is a given $\underline{h}^T$ filter with the corresponding row of $\underline{Y}$ given by the row vector $\underline{u}^T$ output encoding. The use of K multiple SDFs (h_1 to h_K) with different output codings $\underline{u}_k$ or the analogous associative memory can thus be used to distinguish different versions of one class of an object and to discriminate it from other object classes. This analogy is most attractive, since the $\underline{h}_k$ filters synthesized above can be modified to allow different distorted versions of one object (e.g. several $\underline{x}_k$ input key vectors) to be associated with the same encoded output (e.g. the same $\underline{y}_k$ recollection vector) which will now denote the class or subset of several input $\underline{x}_k$ key vectors (i.e. all distorted versions of an input can be assigned the same $\underline{y}_k$).

Incorporation of these pattern recognition techniques into associative memory synthesis allows significantly different recollection vector encodings from the conventional unit vector ones to be employed. Incorporation of these new recollection vector encodings and the associated new associative memory synthesis techniques allows the size of the matrix to be significantly reduced and it adds a distortion-invariant property to the associative memory. As we will show, the use of such encoding techniques actually provides improved noise and storage capacity performance over the conventional unit vector HAMs. We note that for the SDF solution, we must be able to invert $\underline{V}$ and thus this technique also requires linearly independent key vectors, or the use of advanced techniques in the synthesis of such filters. We also note that many pattern recognition preprocessing techniques have been described to achieve the necessary preprocessing to provide linearly independent as well as orthogonal key vectors. Many of these techniques are off-line. However, when the associative memory need not be updated, these synthesis techniques are appropriate.

We now consider the analogy between MSE associative memories with linearly dependent key vectors and the typical MSE LDFs used in pattern recognition when the $\underline{x}_k$ are feature vectors. In this case, the LDFs are denoted by $\underline{w}_k$ and the VIP projection values $\underline{w}_k{}^T\underline{x}$ determine the region in a hyperspace in which the input key vector lies and hence determine the class of the input data. We note that there is no assurance that even the training set data will be correctly classified by this technique (since this is an approximate rather than an exact solution).

Various LDF techniques to calculate associative memories are now summarized. In each case, we calculate a LDF $\underline{w}_j$ and use it as a row of our associative memory matrix. We design this LDF to yield an output of "1" for certain classes and an output of "0" for the other classes (i.e. according to the coding desired and required for that row of the matrix). A multi-class problem is addressed by specifying two classes for each LDF, with each of these two classes being subsets or groups of more than two classes, with the output K-tuple or binary code allowing the final one-of-many class decision to be made. Use of such techniques allows the application of associative memories to image analysis, distortion-invariance, and can significantly increase associative memory storage capacity, as we will note and quantify. LDFs that can be calculated using the training set in-class and between-class scatter matrices include the Fisher LDF and the Hotelling LDF.

3. NOISE PERFORMANCE AND STORAGE CAPACITY OF ASSOCIATIVE MEMORIES

3.1 INTRODUCTION

This section provides a theoretical analysis of noise performance and storage capabilit
and HAMs. We emphasize the difference in AAM and HAM results, the need for a new p
measure and how different recollection vector choices significantly improve results. F
reading, details of several important recent results are included in appendices. Th
emphasize the key points with a minimum of mathematical digression. We first define t
and introduce our notation. The input key vector is $\underline{x} = \underline{x}_k + \underline{n}$, where $\underline{x}_k$ is one of the st
and $\underline{n}$ is a noise vector of zero-mean noise with a covariance matrix $\underline{\Gamma} = \sigma_i^2 \underline{I}$. We
variance of the input and output noise by σ_i^2 and σ_o^2, where the variance of a random v
$\sigma_x^2 = E\{x^2\}-E\{x\}^2$. For zero-mean data, $\sigma_x^2 = E\{x^2\}$. This is the case for the input
noise, since the associative memory matrix operator $\underline{M}$ is linear (if no output thresholdin
We use subscripts to denote specific vectors in a set and superscripts to denote the ele
vector. In this notation, $\sigma_i^2 = E\{(n^i)^2\}$ (from the definition of $\underline{n}$) and $\sigma_o^2 = E\{(y^i-y_k$
requires two terms since $\underline{y}$ is not yet known), where $\underline{y} = \underline{M}\,\underline{x}$, the recollection vector $\underline{y}_k$ c
to the key vector $\underline{x}_k$, and the expectation operation is over only the elements of the vecto
all vectors $\underline{y}_k$.

3.2 PRIOR RESULTS

The typical associative memory performance measure used has been σ_o^2/σ_i^2, where sma
this parameter indicate good performance. Kohonen [1] proved for AAMs that

$$\sigma_o^2/\sigma_i^2 = M/N$$

and reasoned that the result for HAMs would be about the same. Other work [16] showed
incorrect. The documentation of this work is very terse and thus it merits more details,
provide. All steps are provided in appendices, with the results highlighted here. M
simulations were performed for the AAM case [16], with the key vectors chosen from
distribution between -1 and +1 and with the key vectors required to be linearly independe
found to be a requirement, although it is not noted in the original work). The key test ve
formed by adding a zero-mean random variable (with uniform distribution over -1 to +
element of one of the random reference key vectors. For each associative memory matrix
key vectors (each of length N) were tested using one reference vector with ten differ
realizations of noise with the same level σ_i. We assume that this is what was done in t
reference. Different M/N ratios were tested by fixing N = 50 and by varying M ([illegible] te
input vectors used to test each memory matrix $\underline{M}$). For the case of a HAM, each element
element recollection vector was also chosen [16] from a uniform distribution between -1 and
recollection vectors had more than one "1" and are thus not unit vectors).

We define the signal power of a vector to be $E\{(x^i)^2\}-E\{x^i\}^2$. This definition sub
vector's mean from all elements and then calculates the average squared element val

vector. Since the key and recollection vectors were chosen in the same manner in these ear [16], their signal powers are equal and σ_o^2/σ_i^2 is equivalent to the input-to-output SNR ra will use this SNR ratio in our later work (Sections 3.4 and 4) as a preferable performance me more practical cases when the input and recollection vectors do not have the same signal pov proofs of the various theorems to be advanced in this section and in subsequent ones do no equal signal powers for the key and recollection vectors.

We now state four theorems [16]. Proofs of each are provided in the appendices.

- Theorem 1: For any matrix recollection $\underline{y} = \underline{M}\,\underline{x}$, we find $\sigma_o^2/\sigma_i^2 = NE\{m_{ij}^2\}$, wh m_{ij} is an element of $\underline{M}$ and the expectation is over all elements of $\underline{M}$.

- Theorem 2: For an AAM with linearly independent key vectors, we find $E\{m_{ij}^2\}$ M/N^2.

- Theorem 3: For AAMs with linearly independent key vectors, combining Theorems 1 a 2, we immediately find

$$\sigma_o^2/\sigma_i^2 = M/N. \qquad ($$

- Theorem 4: For HAMs, we find

$$\sigma_o^2/\sigma_i^2 = E\{y_{ij}^2\}E\{\mathrm{Tr}(\underline{V}^{-1})\},$$

where y_{ij} is an element of $\underline{Y}$, $\underline{V} = \underline{X}^T\underline{X}$, and the trace (Tr) is the sum of the diagor elements of the matrix noted in parentheses following this operator. The first expect value operator is taken over all elements of $\underline{Y}$ and both expectation operators are tak over the entire ensemble of possible key and recollection vectors.

3.3 DISCUSSION AND ANALYSIS

Theorem 1 is useful since it applies for any matrix with no key or recollection vector assur We will use it in developing more general and more easily evaluated expressions of associative performance.

The result in Theorem 3 agrees with that of Kohonen [1], who obtained his result b different techniques. This result shows and quantifies for linearly independent key vectors (r $M \le N$) that $\sigma_o^2/\sigma_i^2 \le 1$, i.e. an AAM always reduces the input noise (or in the worst case whe N, the input noise is not increased). This also shows that the noise improvement for a AAM i as M/N decreases (i.e. as fewer vector pairs M are stored or when larger dimensionality N key are used). For an AAM design, the amount of input noise expected σ_i^2 is specified and the determines the output noise σ_o^2 one will have to contend with. In later work, we will quan amount of output noise that one can have and achieve a given probability of correct classific different output recollection vector encoding schemes.

Theorem 4 shows that the amount of noise reduction in a HAM depends on the key vect occurs through the Tr($\underline{V}^{-1}$) term) and that it also depends on the recollection vector choic (this occurs through the y_{ij} term) and that its performance does not depend as explicitly on N as is the case with an AAM. This is a most significant result, since AAM storage capacity a performance depends only on M and N. The remark has been made [16] that HAM perform be very poor, even with linearly independent vectors. To see why this might occur, it is nc the determinant of $\underline{X}^T\underline{X}$ can be small (even with linearly independent $\underline{x}_k$ vectors). This occu $\underline{X}^T\underline{X}$ is nearly singular. In this case, Tr($\underline{V}^{-1}$) becomes large and poor performance will res note that poor performance would also result from any associative memory matrix synthesiz the case when $\underline{V}^{-1}$ was hard to compute, i.e. when its condition number was large. We note general AAM performance measure equation does not reflect the effect of the condition num directly. However, the HAM expressions do reflect this issue, through their dependence on matrix $\underline{V}$. Thus, it may appear that HAM performance would be poorer than that performance, even with linearly independent key vectors. However, this is not necessarily the we have noted above. We will quantify these remarks in our data (Section 4). In deriving Th we assume equal energy for all recollection vectors (but their energy is not assumed equal t the key vectors).

We note that the ensemble averages in the equations in Theorem 4 make evaluation of the performance measure for a HAM impossible to evaluate, except by a Monte Carlo techniq Monte Carlo method calculates σ_o^2/σ_i^2 by averaging over a number of different associative r (i.e. different key and recollection vector pairs). For this reason, the results of a Monte Carlo as obtained earlier [16] are not necessarily a good estimation of σ_o^2/σ_i^2 for specific problems other σ_o^2/σ_i^2 expressions are desirable, in which the expectation over the entire ensembl required. In addition, in the prior tests [16], the recollection vectors used were random, h than one "1", and had energy equal to that of the key vectors. This is appropriate for an AA not the conventional HAM situation and (and we shall show) the choice of the recollectio significantly affects HAM performance. Specifically, the test results in [16] are not valid recollection vectors, binary encoded recollection vectors, etc. Also, if the dimensionality of and recollection vectors are different, then the test results in [16] are not too useful. In addi variance of the σ_o^2/σ_i^2 measure can be quite large (especially when averaged over a nu different associative memories). Thus, the resultant σ_o^2/σ_i^2 average can be meaningless a better (smaller) σ_o^2/σ_i^2 values can result for specific HAMs. When the rules we derive f design are used, better σ_o^2/σ_i^2 performance measures will result.

Other σ_o^2/σ_i^2 expressions are possible in the case of unit recollection vectors, $\underline{Y} = c\underline{I}$, wh constant. In this case of HAMs with unit recollection vectors,

$$\sigma_o^2/\sigma_i^2 = (c^2/K)\mathrm{Tr}(\underline{V}^{-1})$$

The second instance in which an equation without all expected value operators is possible oc the case of orthogonal key vectors. In this instance of HAMs with orthogonal key vectors

$$\sigma_o^2/\sigma_i^2 = E\{y_{ij}^2\}Tr(\underline{V}^{-1}),$$ (1

where the expectation operator is the average over all squared elements of $\underline{Y}$. Since c^2 K in equals $E\{y_{ij}^2\}$ for $\underline{Y} = c\underline{I}$, Eq.(10) is equivalent to Eq.(9). Thus, in terms of performance, as unit recollection vectors is analogous to using orthogonal key vectors. This is a notewort result, since one might feel that orthogonal key vectors would yield better performance. Thi follows from linear algebra, since $\underline{V}$ (and $\underline{V}^{-1}$) are diagonal if the key vectors are orthogon yielding only the trace elements of the matrix.

For cases when no conditions on the recollection vectors $\underline{y}_k$ (such as unit recollection vect made and similarly when no conditions on the $\underline{x}_k$ key vectors are made, Theorem 1 can be us alternate σ_o^2/σ_i^2 expression can then be found by substituting Eqs.(A10) and (A13) in the app into Theory 1 to obtain

$$\sigma_o^2/\sigma_i^2 = (1/K)\sum_i\sum_m\sum_k v_{mk}^{-1}y_{im}y_{ik},$$ (1

where v_{mk}^{-1} is the mk-th element of $\underline{V}^{-1}$. Eq.(11) is equivalent to Theorem 1. However, calcu using Eq.(11) are preferable since it provides the result without the need to first explicitly comp

In our quantitative test data, we will use Eqs.(8), (9) and (11) for different cases. Eq.(8) app AAMs with linearly independent key vectors and Eq.(9) applies for HAMs with linearly indep key vectors and with unit recollection vectors. Eq.(10) applies for orthogonal key vectors and has no conditions on the recollection vectors or the key vectors.

3.4 PREFERABLE SNR ASSOCIATIVE MEMORY PERFORMANCE MEASUR

All prior theoretical studies [1,16] of pseudoinverse associative memory noise performanc used the σ_o^2/σ_i^2 performance measure. Other work on associative memory capacity either d consider HAMs, yields bounds (not exact expressions), or does not consider noise. This c performance measure is valid for AAMs, but not for HAMs, since its resultant value can be r (improved) artificially by merely reducing the energy of the recollection vectors (i.e. by usir rather than binary-encoded recollection vectors). Our σ_o^2/σ_i^2 data verifies that unit recol vectors perform better than binary encoded ones. To see the problem with the σ_o^2/σ_i^2 m consider Theorem 1 for the case of a HAM. If we scale each $\underline{x}_k$ by a constant c_k and each $\underline{y}$ constant c_y, then the new associative memory matrix is $\underline{M}' = (c_y/c_x)\underline{M}$, where $\underline{M}$ is the o associative memory matrix. The new expected value (denoted by an apostrophe) is related expected value for the original matrix (denoted by no apostrophe) by $E\{m_{ij}^2\}' = (c_y^2/c_x^2)E\{$

The new and old performance ratios are thus related by $(\sigma_o{}^2/\sigma_i{}^2)' = (c_y{}^2/c_x{}^2)\sigma_o{}^2/\sigma_i{}^2$ From this see that increasing c_x/c_y results in an improved new $\sigma_o{}^2/\sigma_i{}^2$ ratio. However, this improvemen artificial. We note that this issue does not arise for the case of an AAM (since for this matrix recollection and key vectors are the same, and thus have the same energy and scaling factors). T remarks also do not apply to earlier results [16], where equal energy key and recollection vectors used in the Monte Carlo data obtained. This $\sigma_o{}^2/\sigma_i{}^2$ performance could be applied to an HAM $\underline{Y} = \underline{I}$ (or to binary-encoded recollection vectors, or to recollection vectors whose dimensionality K N), by appropriately scaling the recollection vectors, such that their energy and that of the vectors is the same. In general, with arbitrary key vectors and unit or other possible recollec vector encoding schemes, the need exists for a different performance measure.

The performance measure we introduce is the output-to-intput SNR (signal-to-noise) ra SNR_o/SNR_i. The larger this ratio, the better the performance. For equal key and recollection ve energies, this measure and $\sigma_o{}^2/\sigma_i{}^2$ are reciprocals. We define the signal powers as the expected v of the square of the elements minus the square of the expected value of the elements, i.e. we subt off the average or bias energy from our calculations of signal energy. Thus, the signal energies we are

$$s_i{}^2 = E\{|x_k{}^i|^2\} - E\{x_k{}^i\}^2 \tag{12a}$$

$$s_o{}^2 = E\{|y_k{}^i|^2\} - E\{y_k{}^i\}^2, \tag{12b}$$

where the energy values are averages over all elements i of all vectors k. The resultant S performance ratio is then

$$\frac{SNR_o}{SNR_i} = \frac{s_o{}^2\sigma_i{}^2}{s_i{}^2\sigma_o{}^2} \tag{13}$$

For AAMs (with $s_o{}^2 = s_i{}^2$), Eq.(13) reduces to N/M (from Theorem 3) which is the reciproca Theorem 3.

Our concern lies with HAMs. For HAMs with unrestricted key vectors, we combine Eqs.(11) (13) to obtain

$$\frac{SNR_o}{SNR_i} = s_i{}^2 s_i{}^2 \sum_i \sum_m \frac{s_o{}^2 K}{\sum_k v_{mk}{}^{-1} y_{im} y_{ik}}. \tag{14}$$

For HAMs, with $\underline{Y} = c\underline{I}$ (or for the case of orthogonal key vectors), we combine Eqs (10) and (13 obtain

$$\frac{SNR_o}{SNR_i} = \frac{s_o^{\,2}}{s_i^{\,2}E\{y_{ij}^{\,2}\}Tr\{\underline{V}^{-1}\}}. \tag{15}$$

For zero-mean key and recollection vectors, $s_o^{\,2} = E\{y_{ij}^{\,2}\}$ and $s_i^{\,2} = E\{x_{ij}^{\,2}\} = (1/M)Tr\{\underline{V}\}$ U these assumptions, HAMs with $\underline{Y} = c\underline{I}$ (or HAMs with orthogonal key vectors) yield

$$\frac{SNR_o}{SNR_i} = \frac{M}{Tr[V]Tr[V^{-1}]} = \frac{1}{M}, \tag{16}$$

where the last equality holds for orthonormal key vectors, since $\underline{V} = \underline{X}^T\underline{X} = \underline{I}$ and $Tr(\underline{V}) = Tr$ $= M$ for this case. We will employ the different performance measures noted in Eqs.(13)-(15) i quantitative comparison tests of performance in Section 4.

3.5 DATA MATRIX AND PSEUDOINVERSE HAM NOISE PERFORMANCE COMPARISONS

A brief comparison of the data matrix and pseudoinverse HAM with unit recollection vectors ($\underline{I}$) is now provided. Linearly independent key vectors, each normalized to unity, with all elen positive, are assumed. This is necessary for a comparison with no differences in the key vectors, the pseudoinverse HAM requires linearly independent key vectors and the data matrix associ memory requires normalized key vectors. The HAM with $\underline{M} = \underline{Y}\,\underline{X}^+$ and the data matrix with $\underline{X}^T$ are both M × N in size. The data matrix is thus equivalent to a pseudoinverse HAM with $(\underline{X}^T\underline{X})^{-1}$. Thus, in our performance comparison, we compare a HAM with $\underline{Y} = \underline{I}$ to a HAM (the matrix) with $\underline{Y} = (\underline{X}^T\underline{X})^{-1}$. We use Theorem 1:

$$\sigma_o^{\,2}/\sigma_i^{\,2} = NE\{m_{ij}^{\,2}\}, \tag{17}$$

since it applies for any matrix. For the HAM with $\underline{Y} = \underline{I}$ and $M \ll N$, Eq.(17) is most likely less one. For the data matrix, with each row being a normalized key vector, the sum of the squ elements of the matrix rows of $\underline{M}$ is just M, the average squared element is $M/MN = 1/N$ $\sigma_o^{\,2}/\sigma_i^{\,2} = N(1/N) = 1$ from Eq.(17). With Eq.(17) being less for the $\underline{Y} = \underline{I}$ HAM, it will have output noise for a given input noise level. This better performance is expected, since all outpu the $\underline{Y} = \underline{I}$ HAM recollection vector are expected to be zero (except one). To consider how ou noise effects recall accuracy in the two memories, note that all $\underline{Y} = \underline{I}$ HAM outputs are ideally except for the single element with a "1" output; whereas for the data matrix, the non-one ou elements are the vector inner products of the input and the different references and will clearl greater than zero. Thus, the same amount of output noise in each memory can more easily cause matrix output elements to be in error (more easily than is possible for the $\underline{Y} = \underline{I}$ HAM). T differences must be weighed against the advantages of the data matrix HAM, such as: it does require linearly independent key vectors, it yields nearest neighbor performance, it has a large stc capacity (compared to even the HAM with $\underline{Y} = \underline{I}$) and it easily allows the contents of the data m to be altered (by simply changing the vector in one row of the matrix).

4. QUANTITATIVE DATA

This section describes our database, several different associative memories formed, test res associative memories for specific case studies using the different performance measures der Section 3 and the Appendices.

4.1 DATABASE

The database used to provide quantitative test data (versus numerical calculations base theory) for specific pattern recognition problems consisted of 32 × 32 pixel lexicographically binary images of aircraft. Each image was lexicographically ordered into an input key ve dimension $N = 32^2 = 1024$. Two different aircraft, a Phantom and a DC10, were used. The occupied approximately 15% of the full 32 × 32 input image frame. Different images of each rotated in yaw formed different versions of each aircraft for use in different tests and database. The Phantom-18 database contains 18 Phantom jet images at 20° increments in ya a full 360° variation. Our DC10-18 database is similar with DC10 images used. We em Phantom-36 and DC10-36 database set in other tests. These databases contain 36 images pe with 10° increments in yaw now used. We refer to the set of images used to form the memory reference or training set. In some cases, we test the performance of the memory using oth training set images at different yaw rotations. We refer to these as test data. For one HA Phantom and DC10 data are used and the purpose of the associative memory formed is to dist the type of the aircraft, as well as its orientation. In another HAM test, we consider only deter the class of the aircraft, and not its orientation. For noise tests of σ_o^2/σ_i^2 and SNR_o/SNR_i, zero-mean Gaussian noise with five different standard deviations σ_i to the reference Phant image. For each input test image with a given σ_i or SNR_i, we form 10 different input (different input test vectors) with the same σ_i value and SNR_i value (however using 10 di realizations, different seed values, for the given σ_i input noise level). In all noise tests, noisy images were not rebinarized. This allows a better comparison between theory and tests. To p model certain real time optical spatial light modulators, we should rebinarize the noisy input However, we feel that the results obtained with gray-level input test vectors would be represe of those obtained using rebinarized input key vectors to our associative processors.

4.2 TYPES OF ASSOCIATIVE MEMORIES TESTED

To test and quantify associative memory performance, three different associative memorie considered. For consistent results, all memories employed M = 36 key/recollection input vecto (the Phantom-18 and DC10-18 databases). The AAM was formed from Eq.(2) with $\underline{Y} = \underline{X}$. different HAMs were also constructed. HAM-1 used unit recollection vectors with $\underline{Y} = \underline{I}$ in with a different K = M = 36 element output recollection vector for each of the 36 input i The second HAM-2 tested had N = 1024 and M = 36 (as did all associative memories constr and used only two element (K = 2) output recollection vectors $[1,0]^T$ and $[0,1]^T$ for the Phanto

DC10 inputs respectively (i.e. all 18 Phantom key vectors were assigned the same output recollec vector $[1,0]^T$ with the other recollection vector [0,1] used for all DC10 inputs). Since both Phan and DC10 inputs were used in fabricating the associative memories, they achieve both intra-c recognition (e.g. the recognition of different distorted versions of the same aircraft, i.e. a Phant and inter-class discrimination (distinguishing a Phantom from a DC10). The HAM-2 is appropi for image analysis when the type of object rather than its orientation is desired. This is c different from the HAMs conventionally considered. For all associative memories, we calculated using the IMSL Generalized Inverse Subroutine. All key vectors were found to be line independent. This was verified from a calculation of the condition number $(\lambda_{max}/\lambda_{min} = 183)$ $= \underline{X}^T\underline{X}$, which showed that the rank of $\underline{V}$, which equals the rank of $\underline{X}$, was $M = 36$ pseudoinverse thus equals $\underline{X}^+$ in Eq.(3).

4.3 ASSOCIATIVE MEMORY TEST RESULTS USING THE σ_o^2/σ_i^2 MEASURE

Our initial test results are summarized in Table 1. Each entry in this table is the average of realizations of noise with the standard deviation listed. The performance measure tabulate σ_o^2/σ_i^2 for the AAM and the two HAMs constructed. The average of the measured σ_o^2/σ_i^2 value all 50 noise image tests for each associative memory are given in the bottom of the table. theoretical value for the AAM is calculated as M/N from Eq.(8) and it agrees quite well, within with the measured average. For both HAMs, theory and experiment also agreed quite well (wi 1.5% and 11%). The theoretical values for HAM-1 (with unit recollection vectors) were calcul from the trace of $\underline{V}^{-1}$ in Eq.(9) with $c = 1$ and $K = M = 36$. For the second HAM with only K output elements, we calculate the theoretical value using Eq.(11). Several initial obvious remark in order. First, we note general good agreement between theory and tests. Secondly, we note HAM-1 performance is 50% better than that of the AAM (the lower σ_o^2/σ_i^2 performance meas indicate better performance).

The results (for the specific key and recollection vectors chosen) are quite different from other Monte Carlo results averaged over different HAMs (using random key and recollection vect These prior results precicted average HAM performance to be worse than that for AAMs by at 10% when $M \geq 0.2N$. Our final comments concern the performance of the two HAMs. The se HAM (with only two output recollection vectors and two recollection vector elements) perfo worse. This occurs since this matrix is 2 × 1024 with its first row being a sum of the first 18 ro the first HAM and its second row being a sum of the second 18 rows of the first HAM. Recall the size of the first HAM-1 is 36 × 1024. In this case, summing the rows of $\underline{M}$ increases $E\{m_{ij}^2\}$ causes an increase in σ_o^2/σ_i^2 (and thus poorer performance). In general, summing the rows o first HAM will not always increase $E\{m_{ij}^2\}$, since the elements of $\underline{M}$ are bipolar. Here, an inc occurred, because the key vectors corresponding to the added rows are members of the same (rotated yaw views of the same aircraft) and are thus similar, causing the added rows to be sin We discuss these results and preferable performance measures later in Section 4.4.

TABLE 1: σ_o^2/σ_i^2 for AAM and HAM

σ_i	$\frac{\sigma_o^2}{\sigma_i^2}$		
	AAM	HAM $Y = I$	HAM $[1,0]^T, [0,1]^T$ outputs
0.2	0.0352	0.0220	0.0949
0.3	0.0359	0.0218	0.153
0.4	0.0400	0.0253	0.0949
0.5	0.0323	0.0180	0.201
0.6	0.0387	0.0236	0.0655
average	0.0364	0.0221	0.122
theory	0.0352	0.0218	0.136

4.4 ASSOCIATIVE MEMORY TEST RESULTS USING THE SNR_o/SNR_i MEASURE

We now test and compare our three associative memories using our SNR ratio performa measure. Our results are shown in Table 2. Larger values for this performance measure indic better performance. In each case, the data presented is the average of 50 runs for five different no σ_i values, with the measured data obtained from image domain tests. These measured data are th compared to the associated theoretical equations. The AAM results are the reciprocal of those giv in Table 1. For HAM-1 (with unit recollection vectors), s_o^2/s_i^2 is small and for HAM-2 (with $[1,0$ or $[0,1]^T$ recollection vectors) this ratio is large (since HAM-1 has more zeroes in each recollecti vector). Thus, the SNR performance of HAM-2 is better than for HAM-1 (although its $\sigma_o^2/$ performance was worse). Eq.(13) and Table 1 were used for all theoretical calculations in Table From these specific tests, we find AAM noise performance to be better than HAM noise performan (as one would expect) and that different HAMs (such as those with K = 2 output elements, t number of general classes of the data) are preferable to the conventional HAMs (with $\underline{Y} = \underline{I}$ un recollection vectors with K = M = 36 elements and 36 output unit vectors). This represents a ne result. These quantitative results in Table 2 are not necessarily general trends, but are da dependent as we now discuss.

The performance of an AAM depends solely upon the M and N values. HAM performance depen upon $\underline{V}^{-1}$ with HAM-1 performance depending only upon the diagonal elements of $\underline{V}^{-1}$ (because DC1 are slightly larger than Phantoms, the diagonal elements are not the same) and with HAM performance depending upon all elements of $\underline{V}^{-1}$. Since HAM performance depends upon the k vectors used, no general conclusion on AAM versus HAM performance is possible. However, HA with new (binary) recollection vector coding consistently perform better than HAMs with convention unit recollection vectors. Our theory in Section 3 predicted this (for the SNR ratio perform measure). The presence of the elements of $\underline{Y}$ (recollection vectors) in our equations in Sectio confirms this theoretically and our test data in Table 2 quantify it. As discussed, s_o^2, s_i^2 is better HAM-2, which is the reason why our new HAM-2 outperforms HAM-1.

TABLE 2: SNR_o/SNR_i for AAM and HAM

	SNR_o/SNR_i		
	AAM	HAM1 Y = I	HAM2 $[1,0]^T$, $[0,1]^T$ outputs
average	27.47	9.14	15.33
theory	28.41	9.26	13.75

4.5 LARGE CLASS PROBLEMS

The concern in associative processors should be large class problems (M large). We now bri consider how AAM and HAM performance varies with M/N. We expect performance to decrease M/N increases. From Eq.(8), we expect AAM performance to reduce linearly as M increases. HAMs, the performance variation with M will depend upon the specific data. Table 3 shows in results obtained. Eqs.(8), (15) and (14) were used for the three associative memories respectively. second database used 36 images of each aircraft at 10° yaw increments and thus represents a larger = 72 class problem. AAM performance is seen to be linear with M and thus reduces by a factor as shown. The reduction for the HAMs is data dependent. From these data, we clearly see that H performance does not degrade as fast as AAM performance and that at M = 72, the performance HAM-2 and the AAM are approaching each other. Again, this result is not a general trend that can always be assured of (since HAM performance is data dependent). However, this lends furt justification for attention to HAM storage capacity and noise performance and to different out recollection vector encoding schemes.

TABLE 3: Associative Memory SNR_o/SNR_i Performance as M Increases

		TYPES OF ASSOCIATIVE MEMORY		
DATABASE	M	AAM	HAM-1 ($\underline{Y} = \underline{I}$)	HAM-2 $\underline{y}_k^T$ = [1,0] and [0,1]
Phantom-18 DC10-18	36	28.4	9.26	13.75
Phantom-36 DC10-36	72	14.2	6.56	11.84

5. ASSOCIATIVE MEMORY UPDATING

Brief remarks are now advanced on updating (adding, deleting and reassigning key/recollecti vector pairs) in associative memories. We now use subscripts to denote the number of vector pai stored. In the case of an associative memory formed from M key/recollection vector pairs, $\underline{M}$ = $\underline{Y}_M \underline{V}_M^{-1} \underline{X}_M^T$. In this case, we can add a new key/recollection vector pair and calculate the new matrix from the new $\underline{V}_{M+1}^{-1}$. This is possible directly from $\underline{V}_M^{-1}$, by the bordering algorith Extensions of this algorithm allow a vector pair to be deleted. This deletion is easiest if the last $\underline{x}$ and $\underline{y}_M$ vectors are the ones to be removed. To delete another vector pair, we first make this the la vector pair. When the VIP associative memory synthesis technique is used, the key vectors a orthonormal (this can be achieved on-line as we have discussed elsewhere), and updating of the matr $\underline{M}$ is very simple. In a VOP associative memory, the addition of a vector pair is achieved determining the amount of each vector that is new (orthogonal to the prior vector pairs) and includi it (as a VOP, etc.) to the memory matrix. Deleting a vector proceeds similarly, but requires that t vectors be removed from all prior vectors. Updating a data matrix memory is trivial, as t associated row is simply replaced, with no concern for the other rows

6. SUMMARY AND CONCLUSION

This paper has advanced various new theories and expressions for associative memories for neur processing. We first noted the different types of associative memories, the key vector assumptio generally made and the fact that many of these assumptions are not necessarily valid. We advanc new on-line VIP-GS techniques to calculate the pseudoinverse memory from an orthogonal basis s We also noted the differences in storage capacity and noise performance (both issues must considered together) for AAMs and HAMs. We advanced a new and preferable performance measu for more general classes of HAMs. We also derived equations which allow the performance of differe associative memories to be computed more easily and without Monte Carlo techniques. Our resu showed that HAM performance depends on the key and recollection vector choice (whereas AA performance depends only upon the values of M and N). We have noted the similarity betwe associative memory synthesis and LDFs as used in pattern recognition. We find HAM performance be quite dependent on a set of recollection vectors, and we offered new associative memory desig with new recollection vectors (with better performance than conventional HAMs), desig incorporating LDF design techniques, and associative memories with increased memory capacity a reduced memory size. Initial results with such memories appear very promising. Initial remarks associative memory updating with several new algorithms were also advanced.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the Air Force Office of Scientific Research (Gr AFOSR-84-0293) for support of this work. We also acknowledge the assistance of Professor B V Vijaya Kumar for help in deriving the proofs noted in the appendices.

REFERENCES

1. T Kohonen, *Self-Organization and Associative Memory*. Springer-Verlag, Berlin 1984

2. B. Montgomery and B.V.K. Vijaya Kumar, "An Evaluation of the Use of the Hopfield Neural Network Model as a Nearest-Neighbor Algorithm", Applied Optics, Vol 25, 15 November 1986.

3. D Casasent and B. Telfer, "Distortion-Invariant Associative Memories and Processors", Proc SPIE, Vol. 697, August 1986.

4. D. Casasent, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence", SPIE, *Advanced Institute Series on Hybrid and Optical Computers*, Vol 634, pp 439-456, Leesburg, Virginia, March 1986.

5. D. Psaltis and J. Hong, "Shift-Invariant Associative Memories", Optical Engineering, Vol 26, pp. 10-15, January 1987.

6. S.Y. Kung and H.K. Liu, "An Optical Inner-Product Array Processor for Associative Retrieval", Proc. SPIE, Vol 613, January 1986

7. J.A Anderson, "Cognitive and Psychological Computation with Neural Models", IEEE Trans Systems, Man and Cybernetics, Vol SMC-13, No 5, pp 799-815, September October 1983.

8. K Nakano, "Associatron - A Model of Associative Memory", IEEE Trans Systems, Man and Cybernetics, Vol SMC-2, No 3, pp 380-388, July 1972.

9. J Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc Nat Acad Sci USA, Vol. 79, pp. 2554-2558, April 1982

10. D. Psaltis and N. Farhat, "Optical Image Processing Based on an Associative Memory Model for Neural Nets with Thresholding and Feedback", Optics Letters, Vol 10, pp 98-100, January 1985.

11. D Gabor, "Associative Holographic Memories", IBM J Research and Development, pp 156-159, March 1969.

12. P.J. van Heerden, "Theory of Optical Information Storage in Solids", Applied Optics, Vol 10, pp. 665-666, March 1971.

13. B. Soffer, G. Dunning, Y. Owechko, and E Marom, "Associative Holographic Memory with Feedback Using Phase-Conjugate Mirrors", Optics Letters, Vol 11, No 2, pp 118-120, February 1986.

14. G Golub and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions" Numerische Mathematik, Vol 14, No 4, pp 403-420, 1970

15 D Casasent and W T Chang, "Correlation Synthetic Discriminant Functions" Applied Optics, Vol 25, pp 2343-2350, 15 July 1986

16 G S. Stiles and D.L Denq, "On the Effect of Noise on the Moore-Penrose Generalized Inverse Associative Memory", IEEE Trans Pat Anal and Mach Int, Vol PAMI-7 No 3, pp 358-360, May 1985.

17 C R Rao and S K Mitra, Generalized Inverse of Matrices and Its Applications John Wiley and Sons, New York, 1971

18 A D. Fisher and C.L Giles, "Optical Adaptive Associative Computer Architectures" *Proc. IEEE 1985 CompCon Spring*, IEEE,pp 342-344, 1985

APPENDIX A1: PROOF OF THEOREM 1

The output vector is

$$\underline{y} = \underline{y}_k + \underline{M}\,\underline{n} \tag{A1}$$

Substituting (A1) into the definition of σ_o^2 yields

$$\sigma_o^2 = E\{(\underline{M}\,\underline{n})^{i\,2}\} = \sum_j \sum_i E\{m_{ij} m_{ik}\} E\{n^j n^k\} \tag{A2}$$

Using the property of uncorrelated noise that $E\{n^j n^k\} = E\{(n^j)^2\}\delta_{jk}$ the definition $E\{(n^j)^2\} = \sigma_i^2$ and the independence of σ_i^2 from j and k, we obtain

$$\sigma_o^2 = \sigma_i^2 N E\{m_{ij}^2\} \tag{A3}$$

Dividing both sides by σ_i^2, we obtain Theorem 1. This result is valid for any matrix whose key vectors are of dimension N and not just for the pseudoinverse matrix solution. Writing the squared Euclidean norm of $\underline{M}$, we see [17] that the minimum norm solution is $\underline{M} = \underline{Y}\,\underline{X}^{+}$. It can also be shown that this solution is optimal for uncorrelated noise, and that it minimizes $E\{m_{ij}^2\}$ and also σ_o^2/σ_i^2, (i.e. the SNR ratio for the case of uncorrelated noise).

APPENDIX A2: PROOF OF THEOREM 2

For independent key vectors, the solution in Eq.(2) with $\underline{X}^{+}$ defined by Eq.(3) is valid and thus for an AAM

$$\underline{M} = \underline{X}\,\underline{X}^{+} = \underline{X}(\underline{X}^T\underline{X})^{-1}\underline{X}^T. \tag{A4}$$

The trace of $\underline{M}\,\underline{M}^T$ is

$$\mathrm{Tr}(\underline{M}\,\underline{M}^T) = \sum_i (\underline{M}\,\underline{M}^T)_{ii} = \sum_i \sum_j m_{ij}^2 = \mathrm{Tr}(\underline{M}), \tag{A5}$$

where the last equality follows from the fact that $\underline{M}$ is idempotent ($\underline{M} = \underline{M}^2$) and symmetric ($\underline{M} = \underline{M}^T$). The eigenvalues of an idempotent matrix are 0 or 1. The number of eigenvalues that are 1 is $r(\underline{M})$, i.e. the rank of $\underline{M}$, and the trace satisfies $\mathrm{Tr}(\underline{M}) = r(\underline{M})$. To determine $r(\underline{M})$ for $\underline{M} = \underline{X}\,\underline{X}^{+}$ we first show that $r(\underline{X}) = M$ and that $r(\underline{X}^{+}) = M$. It then follows that $r(\underline{M}) = M = \mathrm{Tr}(\underline{M})$. Thus

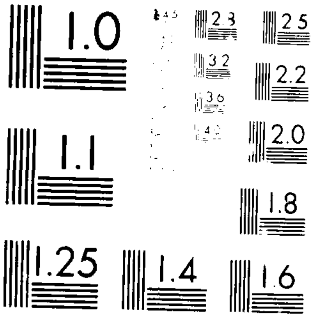
$$\mathrm{Tr}(\underline{M}) = \sum_i \sum_j m_{ij}^2 = M \tag{A6}$$

Using (A6), we prove Theorem 2.

AD-A193 024 OPTICAL DATA PROCESSING(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL AND COMPUTER ENGINEERIN G D CASASENT APR 88 AFOSR-TR-88-0550 2/2

UNCLASSIFIED AFOSR-84-0293 F/G 12/9 NL

END
DATE
FILMED
7 88
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

$$E\{m_{ij}^2\} = \frac{Tr(\underline{M})}{N^2} = \frac{M}{N^2} \qquad (A7)$$

APPENDIX A3: PROOF OF THEOREM 3

This follows directly by substituting (A7) into (A3).

APPENDIX A4: PROOF OF THEOREM 4

We consider Theorem 1, which applies for any matrix and derive an expression for $E\{m_{ij}^2\}$ for the HAM matrix written as $\underline{M} = \underline{Y} \; \underline{V}^{-1}\underline{X}^T$. We first rewrite (A5) for the general HAM case of recollection vectors of dimension K as

$$Tr[\underline{M}\,\underline{M}^T] = \sum_i (M\,M^T)_{ii} = \sum_i \sum_j m_{ij}^2, \qquad (A8)$$

where the summation over i runs from 1 to K and the summation over j runs from 1 to N. To evaluate the Theorem 1 equation for a HAM, we must obtain an expression for $E\{m_{ij}^2\}$. Letting the key vectors $\underline{x}_k$ (of dimension N) and the recollection vectors $\underline{y}_k$ (of dimension K) be random variables, we form the expected value of both sides of (A8) to obtain

$$E\{Tr[\underline{M}\,\underline{M}^T]\} = \sum_i \sum_j E\{m_{ij}^2\}. \qquad (A9)$$

The double summation in (A9) can be rewritten as

$$E\{Tr[\underline{M}\,\underline{M}^T]\} = KNE\{m_{ij}^2\}. \qquad (A10)$$

To evaluate Theorem 1 for this case and hence $E\{m_{ij}^2\}$, we require the trace of $\underline{M}\,\underline{M}^T$

To obtain this, we substitute Eqs.(2) and (3) for an HAM into $\underline{M} \; \underline{M}^T$ and find

$$\underline{M}\,\underline{M}^T = \underline{Y}\,\underline{V}^{-1}\underline{Y}^T. \qquad (A11)$$

The diagonal elements of the matrix product in (A11) are

$$(\underline{M}\,\underline{M}^T)_{ii} = \sum_m \sum_k v_{mk}^{-1} y_{im} y_{ik}, \qquad (A12)$$

where both summations are over the M vector pairs. The trace is the sum of (A12) over the diagonal elements (i = 1 to K) yielding

$$\mathrm{Tr}(\underline{M}\,\underline{M}^T) = \sum_i \sum_m \sum_k v_{mk}^{-1} y_{im} y_{ik} \tag{A13}$$

To evaluate (A9) and hence σ_o^2/σ_i^2, we form the expected value of both sides of (A9) and move the expected value operator within the summation as in (A9). With statistically uncorrelated key and recollection vectors, v_{mk}^{-1} and $y_{im}y_{ik}$ have no cross-correlations and the expected value of their product is the product of their expected values. In practice, this assumption is not realistic, since the $\underline{y}_k$ depend upon the $\underline{x}_k$ and are thus correlated (except for the case $\underline{Y} = \underline{I}$). In tests in [16] each element of each $\underline{y}$ was chosen at random for the data that they used. Thus, $E\{y_{im}y_{ik}\} = E\{y_{im}^2\}\delta_{km}$. This result is not valid for binary encoded $\underline{y}_k$ vectors, but is valid for unit recollection vectors. With these assumptions,

$$E\{\mathrm{Tr}[\underline{M}\,\underline{M}^T]\} = \sum_i \sum_m \sum_k E\{v_{mk}^{-1}\}E\{y_{im}^2\}\delta_{km} = \sum_i \sum_m E\{v_{mm}^{-1}\}E\{y_{im}^2\}$$

$$= \sum_m E\{v_{mm}^{-1}\}\sum_i E\{y_{im}^2\}, \tag{A14}$$

where the last equality follows since $E\{v_{mm}^{-1}\}$ is independent of i. The second summation in (A14) is K times the expected value and $E\{y_{im}^2\}$ is independent of m (for the case of recollection vectors with equal power). This yields

$$E\{\mathrm{Tr}[\underline{M}\,\underline{M}^T]\} = KE\{y_{im}^2\}E\{\mathrm{Tr}[\underline{X}^T\underline{X}]^{-1}\}. \tag{A15}$$

Substituting (A15) into (A10) and the result into Theorem 1, we prove Theorem 4

10. OPTICAL RULE-BASED CORRELATION PROCESSORS

Proc. SPIE
Vol. 752
January 1987

"DIRECTED GRAPH OPTICAL PROCESSOR"

Edward J. Baranoski and David Casasent

Carnegie Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

A directed graph processor and several optical realizations of its input symbolic feature vectors and the multiprocessor operations required per node are given. This directed graph processor has advantages over tree and other hierarchical processors because of its large number of interconnections and its ability to adaptively add new nodes and restructure the graph. The use of the basic concepts of such a directed graph processor offer significant impact on: associative, symbolic, inference, feature space and correlation-based AI processors, as well as on knowledge base organization and procedural knowledge control of AI processors. Initial iconic alphanumeric data base results presented are most promising.

1. INTRODUCTION

Hierarchical tree classifiers have long been used in pattern recognition,[1, 2] particularly for non-parametric problems.[3, 4] Much has been written concerning optimization of tree structures using information theory techniques.[5, 6, 7, 8] However, hierarchical structures have many drawbacks.[9] A major problem is that an incorrect decision at a given node can result in misclassification, since subsequent nodes are not designed to accommodate prior classes. Back-tracking through the tree can compensate for this, but at the expense of classification speed.[10] The major problem is the rigid structure of the tree itself, its limited number of interconnections, and its lack of adaptivity. The optimization techniques mentioned in the literature[5, 6, 8] are very cumbersome and require a great deal of processing. This becomes a problem when an additional class has to be appended to the tree. The problem is that the new class must be added as a terminal node of the existing tree,[9] but classification of future objects of this type is penalized since the new node was not fully integrated into the tree structure. To maintain optimization, the tree must be entirely redesigned, using one of the optimization schemes cited above, for each new added node. This report suggests an alternate modeling for large-class classification problems using directed graphs. Our new version of directed graph techniques is very flexible because new classes and restructured graphs can be accommodated easily without penalty. Our proposed algorithm for directed graph construction is ideal for parallel optical architectures that can quickly perform the computationally intensive steps of multiple filter or discriminant function comparisons at each node of the graph. Optical processing is particularly attractive because of its ability to perform many parallel comparisons concurrently.

The outline of the paper follows. Section 2 explains the topic of directed graphs and introduces the terminology used to describe them. Section 3 extends the concepts of a directed graph to model general classification problems. Section 4 outlines our directed graph algorithm and shows its versatility for adaptation and alteration/adaptivity in the construction and use of the graph. Potential methods of handling input object distortions are also presented. Section 5 outlines potential optical architectures to produce feature spaces and to implement the directed graph algorithm in parallel. Section 6 summarizes the findings of this report.

2. DIRECTED GRAPHS

A directed graph (sometimes called a *digraph*) is a collection of nodes or *vertices* v_n, and a collection of *arcs* joining some or all of the vertices.[11] An example of a directed graph is shown in Figure 1. Note that the graph does not have to be symmetrical. The presence of an arc from v_1 to v_2 does not guarantee that an arc also exist from v_2 to v_1. Symmetry between vertices can be accommodated in this structure by explicitly connecting two nodes with an arc in each direction, as shown between v_2 and v_3. Two vertices joined by an arc are *adjacent*. The *indegree* (*outdegree*)[11] of vertex v_n is defined as the number of arcs entering (leaving) v_n. A *loop* is an arc starting and ending on the same vertex, like the one at v_4. A *path* exists between two vertices if one can travel from one to the other along existing arcs, as between v_1 and v_3. The *cardinality* of a path is the number of arcs contained in that path. A path which starts and ends at the same vertex, such as $v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_2$, is called a *circuit*. A graph is *disconnected* if some nodes are not reachable from other nodes. This is the case for vertices v_1—v_5 which are disconnected from v_6—v_8.

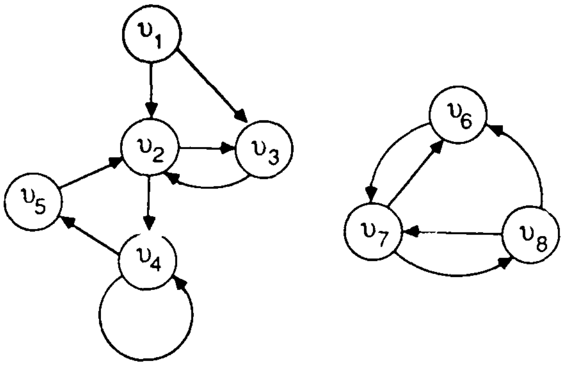


Figure 1: Directed graph

An *adjacency matrix*[11] **A** determines the arcs between vertices, where the element $\mathbf{A}(i,j)$ is equal to one if the graph contains an arc originating from vertex v_i *and ending at* v_j or is equal to zero otherwise. Each row of the adjacency matrix gives the set of adjacent vertices for a given node. The indegree (outdegree) of vertex v_n is equal to the sum of the entries in the n^{th} column (row) of **A**.

A set of matrices $\{\mathbf{A}_n\}$ can be defined where each row of $\mathbf{A}_n$ is the set of vertices that can be reached by paths of cardinality n or less. Using this definition, $\mathbf{A}_1 = \mathbf{A}$ describes simple adjacent vertices. Let the operation $\otimes$ denote binary matrix multiplication, calculated as normal matrix multiplication with numerical multiplication and addition being replaced by logical AND and OR operations respectively. Similarly let $\oplus$ denote a matrix logical OR operation. Then:

$$\mathbf{A}_{n+1} = \mathbf{A}_n \otimes (\mathbf{I} \oplus \mathbf{A}), \quad n \geq 1. \tag{1}$$

Simply stated, Eq. (1) states that v_j is reachable from v_i with a path of cardinality n or less if either $\mathbf{A}_{n-1}(i,j)$ is one or $\mathbf{A}_{n-1}(i,x)$ and $\mathbf{A}(x,j)$ are both one, i.e., a path of cardinality $n-1$ or less must exist from v_i to v_x and an arc from v_x to v_j must also exist. Since all problems are finite, meaning that the size of **A** is finite, a stable result ($\mathbf{A}_{m+1} = \mathbf{A}_m$) will occur for some finite m. $\mathbf{A}_m$ is called the *extent matrix* **E**: it contains the set of vertices that are reachable from every node by any directional path.

3. DIRECTED GRAPHS FOR OBJECT CLASSIFICATION

A classification space can be modeled as a directed graph by mapping each class to a node in the graph. If a wide discrepancy exists between individual members of a given class, distinct subsets of that class can be mapped to different vertices. (In further discussion the term "class" will be used to define the set of objects represented by a node or vertex, regardless of whether such a set is in reality a subclass of a larger class which is represented by several vertices.) Each vertex has associated with it a data vector, either an image or a feature vector, for the given class. The arcs between vertices are chosen to show the similarity or connectedness between classes. If two vertices are adjacent, the classes they represent should be more similar than two classes represented by non-adjacent vertices. The primary focus of directed graph object classification is to determine **A**. Our primary attention is: to construct such an **A** or graph, its use in pattern recognition, and the role for parallel multi-processor optical systems in such a directed graph knowledge base organization or procedural knowledge or control system.

Object classification is achieved by finding the vertex (node) within the graph which best matches the input data vector. The process could start by comparing the input class to several selected vertices in the graph. The starting vertex is the one which most resembles the input data vector. The data vector is then compared to each of the neighbors of this node. Assuming the starting vertex does not represent the input class, a move is made along the arc to the neighbor vertex which is most similar to the input data vector. The input vector is then compared to each of the neighbor vertices of this new node. This process continues until the vertex being examined is more similar to the input vector than any of its neighbor vertices. If the similarity exceeds a certain threshold, then the input belongs to the class represented by that vertex. If the threshold is not exceeded, this vertex is a *local maxima*. One then continues the search to find other higher maxima (using perturbation, i.e. jumps to other regions of the graph). If every node has been examined and no maxima exceeds the threshold, the input data is viewed as a new class and either a new node (class) is added to the graph or the graph is restructured (depending upon memory limitations).

Searching through a directed graph is very similar to traversing a hierarchical tree classifier. All such algorithms yield the final node much quicker than a breadth-first search of every node. The usefulness of the directed graph approach we discuss is the increased flexibility of its structure compared to that of a tree. Unlike a tree, one can start concurrently at several different places within a graph. In addition, changing the starting node is not just a superficial improvement like jumping to a lower node in a tree. Assuming the graph is connected and that each vertex is reachable, the whole classification space can be searched from any node, which is not the case for a tree. However the order of a graph search can vary significantly, since it is strongly dependent upon the starting node. If a crude estimate can be made about the approximate location of the unknown input class within the graph, starting nodes can be picked in that general neighborhood. This will greatly reduce the search time required to examine the whole classification space. We discuss this in Section 4.4.5 and in Section 6.

A major benefit of the directed graph approach is the ease with which new classes can be included in the graph. Adding a new class to a tree is restrictive, since additional nodes can only be affixed to terminal nodes or leaves of the tree; otherwise the whole tree must be redesigned. The interconnections of a graph, on the other hand, can be extended to incorporate new nodes quite easily. Once a graph is modified to include a new class, classification of objects of that class occurs as routinely as for objects in the original classes. Details of this procedure are given in Section 4.

There are a number of pitfalls of varying severity than can be encountered in a directed graph classifier procedure. These include:

1. Disconnected subgraphs within the graph. This could make proper classification impossible, unless perturbations are included (as we suggest and detail) or unless the interconnection of the graph (as we detail) are designed properly.
2. Vertices within a given subgraph with indegree equal to zero. The problem is that a vertex with an indegree of zero is unreachable from any other vertex and could only be located if it was declared as a starting node. The choice of starting vertices should include some of these nodes. Our graph synthesis method and our perturbation step overcomes this problem.
3. Local maxima. The unknown class is theoretically reachable from the starting node, but is not found due to the presence of local maxima in the maximum-ascent approach. Rather than backtracking, we employ perturbations to overcome this problem.
4. Circuits (cyclic paths) within **A**. A circuit exists whenever a diagonal element of **E** is non-zero, meaning that a node is reachable from itself. Since a maximum-ascent algorithm can never return to the same point while still traveling uphill, a circuit is actually a redundant structure which can never be utilized but could reduce the processing speed of a classifier. For a completely connected graph, circuits are

unavoidable. Since every node is reachable from every other, a parent node must be reachable from its neighbor nodes. This requires the use of many circuits. These circuits should be as long as possible, reserving shorter paths for realizable traversals. Shorter circuits will increase the average search time since they force more useful paths through the graph to be longer in length.

Our directed graph processor: uses perturbation, insures connectivity and reachability and long circuits, and it employs hard decisions (rather than simulated annealing techniques) to overcome these potential problems. A recurring problem in large class searches is local maxima.[9] Our two solutions to this problem are now noted. Backtracking is included in our graph by including a working memory with the prior node (not taken) with the largest correlation. Perturbation is included in our graph algorithm, by allowing jumps to new graph regions or prior high-correlation nodes. We prefer hard decisions to simulated annealing (which allows moves to less optimal nodes to occur with finite probability, depending on the correlations or VIP values obtained) to reduce the search space and search time. The high threshold τ we employ also facilitates correct classification (we adjust τ depending upon the number of image pixels and the amount of noise expected).

For pattern recognition applications requiring distortion invariance, we will generally employ a distortion-invariant feature space, using optically generated features.[12] For high-clutter and multi-object cases, we will utilize optical correlators. When distortion-invariance is required in this latter case, smart correlation filters are utilized.[13] For more advanced problems, symbolic correlators are utilized.[14, 15] We emphasize the general knowledge base structure and interconnection (hence its relevance to associative processors, neural processors, and to procedural knowledge rules as well as implicit declarative knowledge inference machines). We use the general term correlation to refer to the use of the nearest neighbor filters per graph node in a correlator or the use of VIPs on input feature vectors. The use of multi-class SDF feature extraction filters[16] to test the M nearest neighbors per node is not recommended (for this large class case considered) since unknown (untrained) inputs per node can produce erroneous results. Thus, the discriminant vector or filter used per node in the graph is that due to the one class considered at that node (this filter can and in many cases is a single class SDF). This filter choice yields better high-confidence results, which is our goal (versus simulated annealing).

4. CONSTRUCTION AND USAGE OF A DIRECTED GRAPH CLASSIFIER

4.1 PARALLELISM AND MULTI-PROCESSORS

In order to build a directed graph classifier, the outdegree M of each node must be selected. M is often selected depending upon the parallelism possible in the processing architecture. If $M=1$, a search through the graph would be entirely sequential. If the number of nodes is L, the search time would then be on the order of LT, where T is the time required to perform the one correlation at each node.

For cases when $M \geq 2$, the number of nodes which must be searched (M comparisons per node) in an "optimal" complete directed graph is on the order of $(log_M L)$, for $L \gg M$.[10] With M nodes checked at each level, an optimal L-class classifier will require x levels, where $L=\sum_{i=1}^{x} M^i=(M^{x+1}-1)/(M-1)$ nodes, i.e. $L(M-1)=M^{x+1}$. Taking the log_M of both sides gives $log_M L+log_M (M-1)=x+1$. Assuming $M \gg 1$, then $log_M (M-1)=1$ and we find $x=log_M L$. This assumes that the graph is laid out such that every node can be reached by exactly one path of length $log_M L$ or less. A graph which satisfies this condition from any set of starting nodes is very difficult to obtain. A *graph efficiency* $\gamma \leq 1$ shall be defined as the inverse of the factor by which the actual search time exceeds the optimal search time of $log_M L$. Thus,

$$\text{Search time} = O(\frac{1}{\gamma} log_M L) = O(log_{M^\gamma} L). \tag{2}$$

γ is a measure of the interconnectedness within a graph. Large γ is preferable. It is very dependent on the size and structure of the graph, as well as the starting nodes chosen. We expect γ to decrease as L increases. If the decrease is not too rapid, good performance will still result. Graphs with many short circuits generally have poor interconnections and will have low values of γ and longer classification times. Conversely graphs with few short path length circuits will have higher γ values and faster classification times. A trade-off must be reached to allow for sufficient interconnections while keeping the classification speed high.

For a sequential (or single channel processor) system, the processing time at a given node is equal to the time it takes to correlate the input with the node's M neighbors, which is equal to MT. Therefore, the total processing time is $O(\frac{1}{\gamma}MT \log_M L)$. Since L and T are constant, this optimum M is obtained by minimizing the processing time with respect to M for $M \geq 2$. Assuming γ is independent of M, we find the minimum total search time for a sequential one processor system when $M=2$. This result is faster than the prior $M=1$ case.

If a parallel processor (or multi-processor system) which can perform N correlations concurrently is used, the time required per node is $O(nT)$, where n is the lowest integer such that $n \geq (M/N)$ and T is the processing time to perform the N concurrent correlations. The number of nodes which must be searched is still $O(\frac{1}{\gamma} \log_M L)$. The minimum processing time is found by minimizing $\frac{1}{\gamma}nT \log_M L$ with respect to M, which occurs when $M=N$ assuming γ is not a function of M. Therefore, optimal classification speed for parallel multi-processors occurs when the outdegree M of each node is equal to the number of processors (i.e. the number of correlations or node VIPs which can be performed concurrently by the parallel system). We use the term correlation to refer to the operation required at each of the M neighbor nodes. This can be a vector inner product (VIP) for the case of input features and some symbols. It can be a 2-D correlation for the case of iconic (image pixel) input data.

A similar analysis shows that the same value of M also represents the optimal number of starting or initial vertices for a given architecture.

4.2 SELECTION OF M NEAREST NEIGHBORS

The construction of the graph from initial data and the updating of the graph for new data are analogous. For L input data column vectors $\mathbf{x}_i$, their similarity is described by the VIP matrix $\mathbf{R}$ with elements $r(i,j)=E[\mathbf{s}_i{}^T\mathbf{s}_j]$ for $i,j \leq L$. We normalize $\mathbf{R}$ by weighting it by $\mathbf{w}$ to obtain $\mathbf{R}_m=\mathbf{w}^T\mathbf{R}\mathbf{w}$, where $\mathbf{w}$ is a column vector with elements $w(i)=[\sum_{j=1}^{n} s_i(j)^2]^{-1/2}$. Normalization by the difference between the input data vector and the mean data vector is also possible. The weighting by the inverse of the magnitude of the data vector produces $\mathbf{R}_m$ with diagonal elements equal to 1 and all other elements less than one. This presents a vector with a high magnitude from dominating the correlation results[17] while still retaining a positive-definite nature matrix for $\mathbf{R}$. From $\mathbf{R}_m$, one can produce an adjacency matrix $\mathbf{A}$ with elements

$$a(i,j) = \begin{cases} 1 & \text{if } r(i,j) \text{ is one of the } M \text{ largest elements in row } i \text{ of } \mathbf{R}_m,\ i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The provision that $i \neq j$ prevents single node loops in $\mathbf{A}$. The reachable extent matrix $\mathbf{E}$ can then be determined using Eq. (1).

From tests, we find that $\mathbf{A}$ computed from $\mathbf{R}_m$ by Eq. (3) alone yields a well-structured graph of nearest-neighbors, but is not necessarily a well connected graph. This is especially apparent when one considers a multi-class problem where there are $M+1$ very similar classes. Using the above procedures alone, these $M+1$ classes will form an isolated subgraph, unconnected from all the remaining nodes. We thus use $\mathbf{R}_m$ to assign outgoing nodes and a more detailed procedure (detailed below in Section 4.4) to provide incoming nodes and the connectivity of the graph.

4.3 DEFINITIONS

The following definitions will be used in subsequent analysis:

1. L is the number of classes currently represented in the graph;
2. L_{max} is the maximum number of classes (nodes) permissible in the graph. It is upper-bounded by the memory constraints of the system;
3. M is the maximum outdegree permissible for any node; it is determined by the degree of parallelism in the processing architecture (the number of channels which can be processed concurrently);
4. $\mathbf{x}$ is a column vector representing the new original input data;
5. $\mathbf{x}'$ is the normalized data vector for the new input data;
6. $\mathbf{s}_i$, $i \leq L_{max}$ is the normalized data vector (discriminant vector) of class i (i.e. at node i);
7. τ is the acceptable threshold which must be exceeded for a match to occur between the input and a given class;
8. v_c is the node currently being examined;
9. v_L is a new node being appended to the graph;
10. $C(i,j)$, $i \leq L_{max}$, $j \leq M$, is the j-th highest element in the i-th row of R_m;
11. $K(i,j)$, $i \leq L_{max}$, $j \leq M$, is the column number of the j-th highest element in the i-th row of $\mathbf{R}_m$;
12. $I(i)$, $i \leq L_{max}$, is the indegree of v_i;
13. $E(i,j)$, $i,j \leq L_{max}$, is the (i,j) element of the reachable extent matrix;
14. $Z(i)$, $0 \leq i \leq L_{max}$, is an $L_{max}+1$ element work array containing the result of correlations or VIPs of $\mathbf{x}'$ with previously stored classes (represented by $\mathbf{s}_i$).

The matrices $\mathbf{C}$ and $\mathbf{K}$ are actually abbreviated versions of $\mathbf{R}_m$ and $\mathbf{A}$, respectively (containing their largest elements). The i-th row of $\mathbf{K}$ contains the column numbers j where $a(i,j)=1$. Similarly, the i-th row of $\mathbf{C}$ contains the elements of $\mathbf{R}_m$ corresponding to the same locations where $a(i,j)=1$. The $\mathbf{C}$ and $\mathbf{K}$ matrices reduce the storage requirements by a factor of L/M.

4.4 OPERATION

Figure 2 illustrates the basic operation of a directed graph classifier. The input data $\mathbf{x}'$ is normalized and (if required) distortion invariant. An initial threshold $\tau < 1$ is defined to determine whether an acceptable match has been found at each node. We make τ high enough so that distinct classes will not be categorized together and yet not so high that any noise in the input will inhibit proper classification and force the graph to create a new class. With low noise expected, one should set τ conservatively high. Then, even minor deviations in a prior input will cause the graph to think of the input as a new class. As the number of classes grows and approaches L_{max}, the threshold is lowered, similar nodes (classes) are combined and the graph is restructured. This forces a new segmentation of the data. This will enable the classifier to adjust τ to the actual problem set, while controlling the number of nodes in the graph. The input data can be time sequential scenes, objects, or the contents of a knowledge base. Assume that the input will be a sequential stream of class data including noise and possible distortions. The steps of the algorithm for synthesis or use of the graph follow.

4.4.1 Initialization of the Graph

1. Initialize all matrices to zero.
2. Preprocess the first M input data vectors $\mathbf{x}_i$, yielding $\mathbf{x}'_i$.
3. Since we started from a zero-class classifier, these M vectors are stored as the first M nodes $\mathbf{s}_i$ (for $i \leq M$) in the graph. They are used as the initial starting vertices. L is set to M.

4.4.2 Classification of New (Subsequent) Input Data Vectors (Iterations)

This iterative procedure applies in general when the graph contains more than M nodes.

1.
   a. Preprocess the input data vector to yield $\mathbf{x}'$.
   b. Correlate $\mathbf{x}'$ with each of the starting vectors in the graph. Set the current node v_c to the vector with the highest correlation with $\mathbf{x}'$, and store the correlation as $Z(0) = max[\mathbf{x}'^T\mathbf{s}_i]$, for all $i \leq M$. $Z(0)$ is the current maximum correlation.

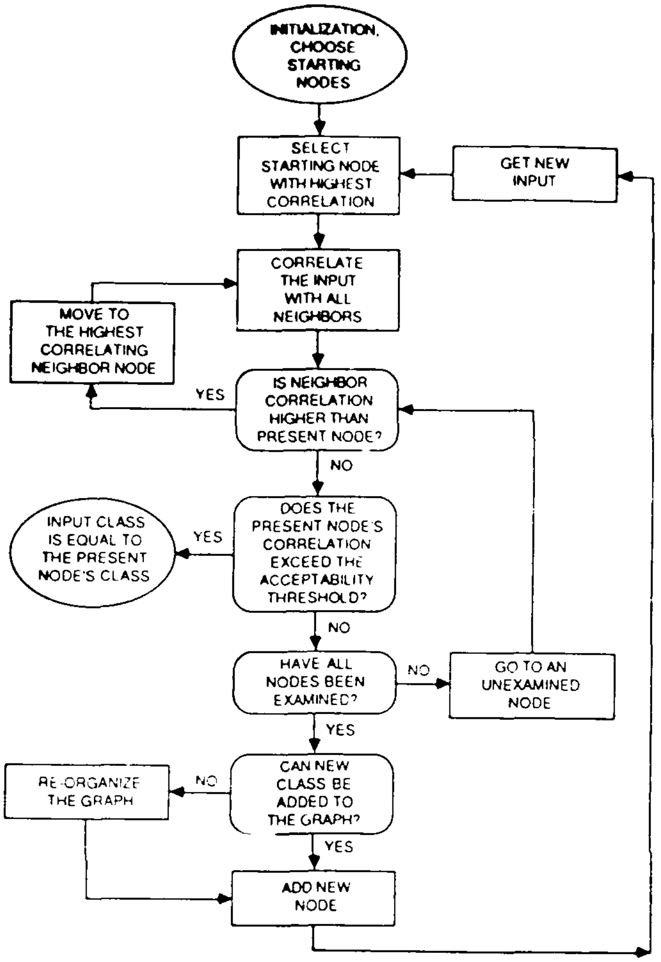


Figure 2: Block diagram of a directed graph classifier

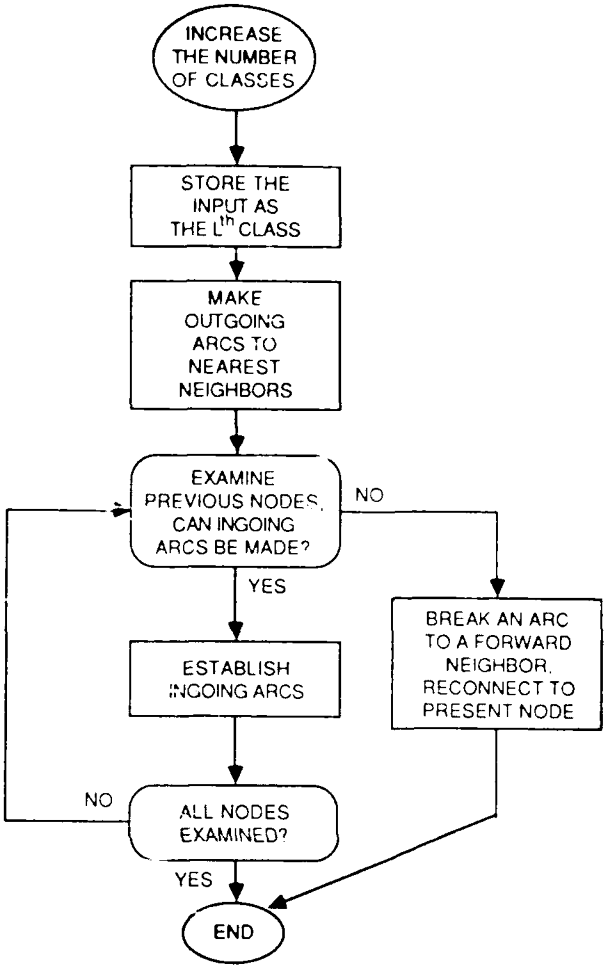


Figure 3: The addition of a new node to a directed graph classifier

c. Correlate $\mathbf{x}'$ with the M neighbors of v_c, found in the matrix $\mathbf{K}$. Store these results in $Z(K(v,i)) = \mathbf{x}'^T \mathbf{s}_{K(v,i)}$, for all $i \leq M$. These calculations are not excessive. Some of the neighbors of the current node could be neighbors of previously searched nodes, in which case their correlations would already have been calculated and stored in $\mathbf{Z}$. Recalculation of them is not necessary.

d. Look for the highest correlation among the neighbors of v_c If this is greater than $Z(0)$ then set $Z(0)$ equal to it, set v_c to that node, and repeat step c.

e. At some point $Z(0)$ is greater than the correlation at any of the neighbor nodes. If $Z(0) \geq \tau$, then the input is classified as belonging to the class represented by v_c. Classification of the input is now complete and the next input vector can be classified.

f. If $Z(0) < \tau$, we recognize v_c as a local maxima of the graph. In the case of construction of the graph, we examine all nodes, using backtracking or perturbations (to new graph areas or to prior nodes with a high Z, i.e. perturb or jump by backtracking). We now briefly discuss three techniques[9] to avoid being trapped in a local maxima. They generally apply to use of the graph, rather than construction of it.

<u>Back-tracking</u>: This involves going back to a previous node and taking an alternative route. This technique can avoid searches for poor solutions.

<u>Perturbation</u>: This technique permits random jumps to unsearched nodes of the graph.

Simulated annealing: This is a non-deterministic searching process which allows "downhill" rather than "uphill" moves to occur with a small (but finite) probability, depending on the ratio of the data vector's correlation with v_c and each of the neighbors of v_c.

In operation, we prefer (in order of preference) to: (1) jump to the next largest starting node (if its correlation is close to that of initial node chosen), (2) jump to an alternate neighbor of a prior node, or (3) perturb to unexamined areas of the graph.

These searching techniques in steps (a) to (f) continue until a match is found or until every node in the graph has been searched. This procedure is much faster and easier than might appear. The number of steps required (and hence the number of nodes searched) is $O(log_M L)$ and the memory is $O(L)$. If the entire graph is searched and no correlation exceeds r then a new node must be added to the tree. The procedure is outlined in Section 4.4.3.

4.4.3 Addition of a New Node

This step outlines how a graph can be modified to include a new class. A block diagram of the procedure is shown in Figure 3. Its steps follow.

1.
   a. Increment L, the number of classes stored in the graph, by one. If $L > L_{max}$, reorganize the graph as in Section 4.4.4. If not, proceed as below.
   b. Store $\mathbf{x}'$ in $\mathbf{s}_L$. This is the data vector for the new class, which will be represented by v_L in the graph.
   c. Add M outgoing arcs from v_L. If $Z(i)$ is the j-th highest element $(1 \leq j \leq M)$ in $\mathbf{Z}$, then set $C(L,j) = Z(i)$ and $K(L,j) = i$ and increment $I(i)$ by one. This establishes arcs emanating from v_L to its M closest neighbors as set by $\mathbf{R}_m$. These new neighbors will be referred to as forward neighbors. This establishes the outdegree of v_L as $min(L,M)$.
   d. Establish ingoing arcs to v_L. This step requires certain precautions to maintain connectivity and reachability. We require that every node have a non-zero indegree. This implies that the sole ingoing arc to some node v_i cannot be broken to establish an arc to v_c unless v_c in turn has v_i as a forward neighbor, re-establishing connectivity to v_i. This will force the graph to be connected, while also preventing subgraphs. We achieve this in an ordered manner as follows.
      i. Check all previous nodes to see if an arc should connect any of them to v_L, i.e. if v_L correlates well with a prior node (better than some prior arc). To retain the graph's symmetry, this requires that $Z(0) > C(i,M)$ for some v_i. To guarantee connectivity, v_i must still be reachable from v_L without the arc $v_i \rightarrow v_{K(i,M)}$ (i.e. another way must exist to reach the node whose ingoing arc was broken from v_i). Reachability is found using a modified $\mathbf{A}$ matrix where $a(i,K(i,M)) = 0$.
      ii. If step i returns a positive result for some v_i, the arc connecting v_i to $v_{K(i,M)}$ can be broken and replaced with one connecting v_i to v_L. The reachable extent of v_i and the connectivity of the graph will not be adversely affected. $C(i,M)$ and $K(i,M)$ are changed to $Z(i)$ and i, respectively. The i-th row of $\mathbf{C}$ and $\mathbf{K}$ is now sorted to accommodate the new data. This step is repeated for all v_i which apply.
      iii. If no ingoing arcs to v_L are formed using the above steps, meaning that $I(L) = 0$, we must still force a connection. This is most conveniently done by breaking an arc from some other node that also has an ingoing arc to a forward neighbor of v_L. The forward neighbor with the highest correlation is the best choice. The arc is then reconnected to the new node v_L as outlined in step ii. This will maintain the graph's connectivity at the expense of potential small drops in the graph's classification space when searching for particular classes.
   e. The reachable extent of v_L is stored in the $L-th$ row of $\mathbf{E}$. It is equal to the union of the set of the neighbors of v_L with the set of all nodes reachable from those neighbors. This means it is unity in any column j where $K(L,j) = 1$ or $E(K(L,k),j)$ for any $k < L$.

4.4.4 Reorganization of the Graph

If L exceeds L_{max}, the graph has outgrown the algorithm. The threshold τ must be lowered so that new classes are not encountered as frequently and such that old prior classes can be merged. The following procedure lowers L by one node, merging several prior nodes and reorganizing the graph, while still retaining the graph's connectedness It can be used repetitively until $L \leq L_{max}$:

1.
    a. τ should be lowered so that it is equal to the highest value of the first column of **C**.
    b. Merge the node v_i, which satisfies $C(i,1)=\tau$, with node $v_{K(i,1)}$. The data vectors of these two nodes can be averaged together to create a new discriminant vector representative of the two merged classes.
    c. All arcs to v_i and $v_{K(i,1)}$ are broken and replaced by arcs to other existing nodes. This step is equivalent to removing the i-th and $K(i,1)-th$ columns of both **A** and **R**. This could potentially effect the connectivity of the graph. If this occurs, the replacement arcs should be chosen so that the connectivity is re-established.
    d. The indegrees of all the forward neighbors of v_i and $v_{K(i,1)}$ are reduced by one. This removes the i-th and $K(i,1)-th$ columns of C and K. At this point, both v_i and $v_{K(i,1)}$ are removed from the graph.
    e. The merged node is now added to the graph using the procedure outlined in Section 4.4.3.

4.4.5 Multiple Initial Starting Nodes and Meta-Vertices

To improve the connectivity and reachability of all nodes in the graph, meta-vertices can be established. These vertices are not class nodes, but are used to connect subgraphs (isolated from the graph). These nodes slow processing and search time and are avoided in our graph synthesis algorithm. We mention them as a possibility for severe cases.

At the initial input to the graph, we enter the graph at M points (since we have M processors, we use them at all levels, i.e. at the initial level also). For this case, meta-vertices are useable (or other key or parent vertices) as some of the initial choices for the M starting initial nodes.

5. OPTICAL IMPLEMENTATION

Optical architectures are very appealing for this algorithm since they can easily perform the feature extraction and required correlation operations in parallel. One architecture to achieve the M correlations (or VIPs required per node) in parallel is shown in Figure 4. In this figure, the preprocessed input data vector $\mathbf{x}'$ is applied to a single-channel acousto-optic (AO) cell. The cylindrical lens L1 vertically replicates the data vector across the correlation plane where a spatial light modulator (SLM), such as a multi-channel AO cell, is placed. The spatial light modulator contains one data vector on each of its rows. The projection of $\mathbf{x}'$ onto each of these rows produces the point-by-point product of every component of $\mathbf{x}'$ with the corresponding components of the data vectors stored in the SLM. Another cylindrical lens (L2) sums these products across each row, producing the vector inner product (VIP) of $\mathbf{x}'$ with each data vector stored in the SLM. L2 focuses the correlation results on a linear detector array, where they are fed to an external controller.

The controller is responsible for loading the SLM with the necessary data vectors to traverse the directed graph It initially loads the SLM with the starting vertices of the graph. It then detects the highest output and assigns v_c as that node. The neighbors of that node are loaded into the SLM, and the process continues until the input is classified as either an existing class or a new class which must be accommodated in the graph. Other optical architectures (such as ones with input point modulators, a one-channel AO cell, and N 1-D time integrating detector arrays are also viable alternatives). Variations of each system to allow high-accuracy encoded data processing are also possible. In Figure 4, one would input an encoded description of each element, perform a high-accuracy multiplication (by convolution), and continue for the next vector element.

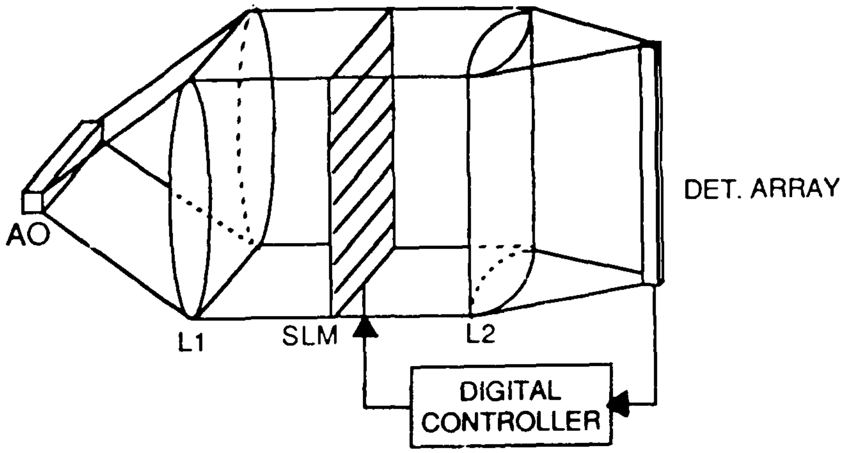


Figure 4: Example Architecture for an Optical Directed Graph Classifier

The hybrid architecture of Figure 4 and its variations use the best of two different technologies: optics is used to handle the heavy computational burden, while digital memory provides the storage of the data vectors and the graph's **A**, **C** and **K** matrices. Such a system is suitable for very large classification problems as we now quantify.

The key component of this system is obviously the SLM. As shown, maximum classification speed for a parallel directed graph classifier is obtained when M is set equal to the number of correlations which can be performed concurrently. Therefore, M is set by the number of data vectors which can be stored in the SLM. For example, consider a 16-channel AO cell as the SLM, with digital hardware capable of loading the cell at a 16 Mbps rate (1 Mbps per AO channel). This would allow M=16. A 50-long vector with 8-bit resolution for each vector component could thus be passed through each cell in 0.4ms. This will be the time T required to perform the parallel correlations. The controller synchronizes the SLM data with the input data. Since T is greater than the propagation time through the multi-channel AO cell, the system performs time integration in T=0.4ms per node searched. Assuming a total of 2^{12} classes (L=4,096), the average time for classification would then be $O(T\,log_M\,L)$=1.2ms. Here we see that the penalty for back-tracking is the addition of T (a 30% increase) for each back-tracked step. The digital memory requirement for this example is approximately 0.5 Mbits.

Another alternative is a liquid crystal SLM, which presently offer resolution of about 100×100 at video rates (30 Hz) with 32 grey levels (5 bits/pixel). The processing time T per node is now 33ms, which yields a much slower classification time than the multi-channel AO cell case. Projections have been made for improvements in all of these figures, notably an increase in its frame rate to 1 kHz. Such improvements would be necessary to make liquid crystal SLMs feasible for such a system.

6. DIRECTED GRAPH CASE STUDY

The algorithm was tested using standard 5×9 dot-matrix alphanumeric characters in 62 classes ('A' through 'Z', 'a' through 'z', and '0' through '9'). Samples of the characters are shown in Figure 5. Each character was described by a 64-element binary vector, which was obtained by taking each row of the character and making that the next five elements of the data vector. The remainder of the vector was zero-padded. The number of forward neighbors for any node was chosen to be M=4. The graph was built one class at a time, using a threshold τ of 0.99.

Figure 6a illustrates the initial 5 class graph and the resulting graph when the sixth node ('F') was added to the five-node classifier. This was done by first adding outgoing arcs from 'F', then by determining what arcs should be broken to make ingoing arcs to 'F'. First outgoing arcs were made from 'F' to the four nearest neighbors which had the highest correlations with 'F' (in this case 'A', 'B', 'D', and 'E'). Next, ingoing arcs to 'F' were established by checking each of the five previous nodes to see if 'F' correlated better than a given node's lowest correlation neighbor. If this was the case for some node and if its lowest neighbor was reachable from 'F', then that arc was

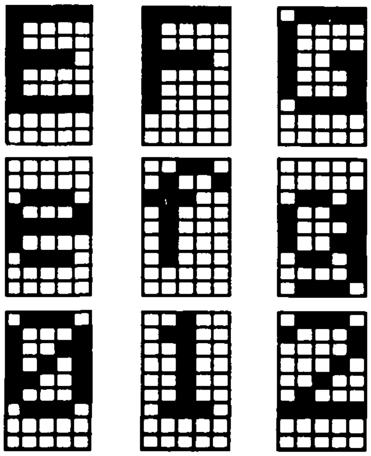
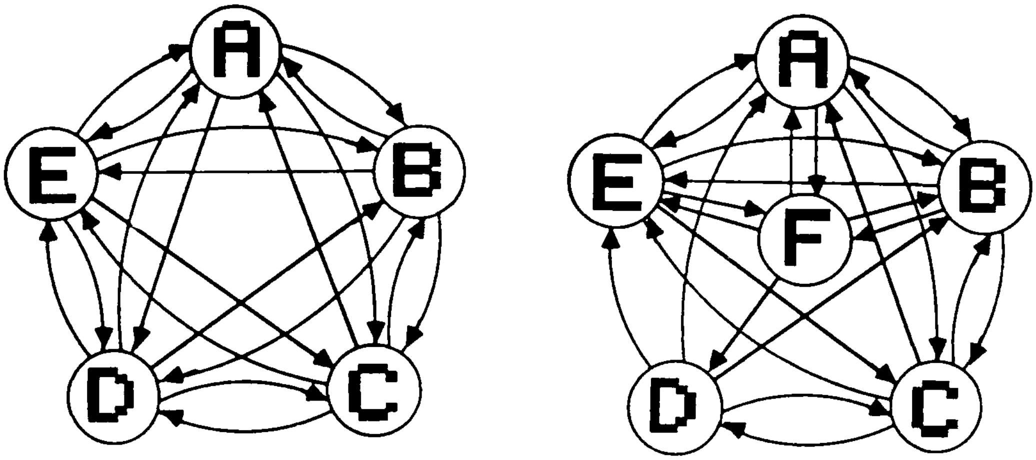


Figure 5: Standard 5×9 Dot-Matrix Alphanumeric Characters



(a) the five-class graph 'A' through 'E' (b) the six-class graph 'A' through 'F'

Figure 6: The Addition of Node 'F' to the Five-Class Graph ('A' through 'E')

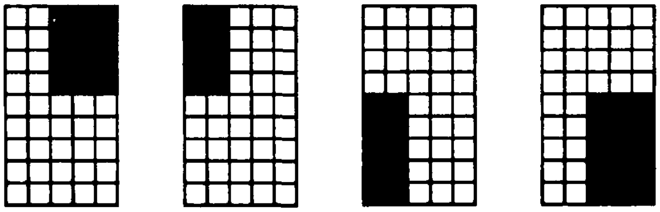


Figure 7: Meta-Vertices Masks used (Number of Pixels Per Quadrant) for Initial Input Node Tests

Table 1: Directed Graph for a Character Data Base

```
    ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
[A]  B     H       P R
[B] A      H       P R
[C]       G                                             0     6 8
[D]               O Q   U                         u
[E]      FG    L   P
[F] AB  E     K
[G]   C E         O                                     0
[H] AB          M                    h
[I]   C                T                                       78
[J]                                                       234  7
[K]        H   LM    R
[L]    DE               U      b
[M]        H     N        W          h
[N]                     U WX            k
[O]   C   G         Q                                   0
[P] AB   F           R
[Q]    D          O                               u     0
[R] A    F         P                                 x
[S]   C                                                       6 89
[T]         I               Y            l               1
[U]               O       W    b d
[V]        H       P    U   Y
[W]        H     N      U                           w
[X]        H     N          Y           k
[Y]                    T VWX
[Z]     E                                               0    5 7
[a]                              de         o   s
[b]                             c    h     n                 5
[c]                            b de         o
[d]                     U     ab              q
[e]                           a c           o   s
[f] A           M                          n                 5
[g]                              d          o q       y
[h]              N             b           n   r
[i]         I                            l      s        1
[j]                                fg         q       y
[k]        H     N         X         h
[l]         I          T              i                  1
[m]                            b     h     n   r
[n]                            b     h    m    r
[o]                            bcde
[p]                            b           no  r
[q]                              d  g             u   y
[r]                                  h    mn p
[s]                             c e         o    t
[t]                                          p  s      z     5
[u]                     U           g         q    v
[v]    D                                          u w y
[w]               O     U W                       u
[x]  B     H         R                                 z
[y]                                 g  j      q   u
[z]                           a       i         s         2
[0]   C           O Q                                           8
[1]         I          T              i  l
[2]                 QR                                 z        8
[3]       G           S                                      5  8
[4]          J                a   e             s
[5]       G       O            b                           3
[6]     E G           S                                         8
[7]          J         T     Z                          0
[8]               O   S                                 0     6
[9] A               Q S                                         8
```

broken and replaced with an arc to 'F' (Figure 6b). Table 1 shows the adjacency matrix **A** (with its elements noted) for the actual graph obtained. Each row of the matrix shows the neighbors for the particular character in the left margin.

A meta-vertex was used at the starting node. It consisted of the four masks shown in Figure 7, which simply counted the number of "on" pixels in each quadrant. For this problem the optimum number of nodes to be examined (on the average) for classification is $(16\times2+(62-16)\times3)/62=2.74$, where examining a node refers to examination of its $M=4$ nearest neighbors. This value is simply the average of the path lengths given an "optimal" graph. The actual value obtained in tests on these data was 5.27, yielding the graph efficiency $\gamma=0.52$. While the efficiency may seem low for this particular example, one should remember that γ reflects the interconnectedness in the graph, which is achieved at the expense of some classification speed. More research is required to determine the effects of various system parameters on γ. Without the input initial meta-vertices, performance was much worse (an average of about 8.5 nodes per search).

7. CONCLUSIONS

An algorithm has been presented to model a large-class problem or large knowledge base as a directed graph classifier. It is shown that the classification procedure can yield the object class quite well. The proposed algorithm can also be used to iteratively synthesize the graph one class at a time, while maintaining the graph to be connected and all classes reachable. Our algorithm allows the classifier to easily accommodate new classes, and it is especially suitable for parallel processing architectures, such as optical systems. Initial results were most promising. This concept appears to have use in many new optical AI concepts.

ACKNOWLEDGMENT

The support of this work by the Air Force Office of Scientific Research and its partial support by the Defense Advanced Research Project Agency is gratefully acknowledged.

References

1. P. H. Swain and H. Hauska, "The decision tree classifier: design and potential", *IEEE Trans. Geosci. Electron.*, Vol. GE-15, No. 3, July 1977, pp. 142-7.

2. J. K. Mui and K. S. Fu, "Automated classification of nucleated blood cellls using a binary tree classifier", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-2, No. 5, Sep. 1980, pp. 429-43.

3. J. H. Friedman, "A recursive partitioning decision rule for nonparametric classification", *IEEE Trans. Comput.*, Vol. COMP-26, No. 4, April 1977, pp. 404-8.

4. E. M. Rounds, "A combined nonparametric approach to feature selection and binary decision tree design", *Pattern Recognition*, Vol. 121980, pp. 313-7.

5. I. K. Sethi and G. P. R. Sarvarayudo, "Hierarchical classifier design using mutual information", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-4, No. 4, July 1982, pp. 441-45.

6. C. R. P. Hartman, P. K. Varshney, K. G. Mehrotra, and C. L. Gerberich, "Application of information theory to the construction of efficient decision trees", *IEEE Trans. Inf. Theory*, Vol. IT-28, No. 4, July 1982, pp. 565-77.

7. Q. R. Wang and C. Y. Suen, "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-6, No. 4, July 1984, pp. 406-17.

8. S. Watanabe, "Pattern recognition as a quest for minimum entropy", *Pattern Recognition*, Vol. 13, No. 51981, pp. 381-7.

9. D. A. Jared and D. J. Ennis, "Learned pattern recognition using synthetic-discriminant-functions", *Proc. SPIE, Vol. 638*, April 1986, pp. 91-101.

10. H. S. Stone and P. Sipala, "The average complexity of depth-first search with backtracking and cutoff", *IBM J. Res. Develop.*, Vol. 30, No. 3, May 1986, pp. 242-58.

11. N. Christofides, *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975.

12. D. Casasent, "Hybrid Optical/Digital Image Pattern Recognition: A Review", *Proc. SPIE, Vol. 528*, Jan. 1985, pp. 64-82.

13. D. Casasent and W.T. Chang, "Correlation Synthetic Discriminant Functions", *Applied Optics*, Vol. 25, July 1986, pp. 2343-2350.

14. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations", *Proc. SPIE, Vol. 625*, Jan. 1986, pp. 220-225.

15. D. Casasent and A. Mahalanobis, "Rule-based, probabilistic, symbolic target classification by object segmentation", *OSA Topical Meeting*, March-April 1987.

16. D. Casasent and H. Okuyama, "A high-dimensionality pattern recognition feature space", *Proc. SPIE, Vol. 579*, Sep. 1985, pp. 245-257.

17. J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, J. Wiley & Sons, New York, 1965.

11. OPTICAL DIRECTED GRAPH PROCESSORS

Rule-based symbolic processor for object recognition

David Casasent and Abhijit Mahalanobis

The application of symbolic processing and rule-based methods for target recognition using correlation filters is considered. The concept of partitioning images is introduced, and its advantages are described. Techniques for rule development, symbolic substitution, error correction via associative processing, and on-line filter adaptation are advanced. Initial simulation results are also presented and discussed.

I. Introduction

A. Background

The use of spatial filters for the automatic recognition of targets has been widely studied. Typically, such filters are synthesized to recognize complete objects. In this paper, we address the possibility of identifying targets by parts (i.e., by partitioning the input image), and by symbolically analyzing the partitions simultaneously.

The fundamental idea is to generate a symbolic description of the input image using spatial filters (also referred to as correlation filters).[1] Separate filters are synthesized for different spatial regions of the composite set of training images. A composite filter of all objects (with the spatial relationship between segments preserved) is formed. It is then correlated with the input to obtain a symbolic or multibit code description of the input object. The K-tuple synthetic discriminant function (SDF) investigated in previous research[2] also yields a multibit output code. However the prior K-tuple SDF differs from the scheme proposed in this paper in one important aspect. Unlike the correlation filters employed for symbolic processing, the prior K-tuple filter systems are synthesized from entire training images. The advantages of our new proposed scheme will be discussed shortly.

Some relatively simple 3-D objects such as aircraft can be numerically modeled on a computer.[3] Most aircraft are a collection of generic parts whose dimensions differ from model to model. Computer algorithms can efficiently generate the images of most aircraft parts and combine them to produce realistic images of existing civilian and military aircraft. This is possible mainly because the number of aircraft parts is small, because aircraft have a consistent set of generic parts and because they can be modeled by simple geometric shapes such as cones, cylinders, and planes. Hence correlation filters can be synthesized for various aircraft parts, and the target class be identified on the basis of those parts which are visible in the input image.

A category of objects such as tanks is more difficult to model because the number of variations in structure, shape, and size is very large. Computer programs for modeling tanks exist[4] but result in very specific models for each tank. It is difficult to obtain images for individual tank parts from computer models and thus correlation filters synthesized from tank parts are not easy to assemble. In this paper, we propose an alternative scheme based on spatially partitioning training set images that serves the same purpose of recognition by parts for more complicated objects such as tanks.

B. Practical Motivation

As stated in Sec. I.A, it is conventional to synthesize distortion-invariant linear combination correlation filters from complete training images. However, problems may arise when parts of the object are absent or invisible either due to occlusion by artifacts in its natural environment (such as foliage, terrain, camouflage measures), noise in the input, temperature variations when an infrared imaging sensor is used, sensor malfunction, and a host of other possible reasons. In situations where the entire target is not visible, it is preferable to identify its observable parts and from these logically deduce its class. Analogously, one can determine the more reliable parts of the object and give them more weight than other parts. Our proposed symbolic processor is motivated by this set of practical considerations.

The inference of object class from a study of the visible object parts requires "abductive reasoning."[5] Formally speaking, abductive reasoning involves the establishment of pertinent facts to infer a new fact. Since more than one answer is often possible, abductive reasoning must also yield which answer is the best. To make decisions of this nature, we must weigh the available evidence. To do this, we must know how strongly a fact weighs for or against a conclusion, and how to combine the pieces of evidence into a final conclusion. To gain evidence, it is necessary to obtain prior and conditional (*a posteriori*) probabilities. A technique to achieve this will be discussed in futher detail in Sec. V.

An expert system is often defined as a rule-based

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 27 March 1987.
0003-6935/87/224795-08$02.00/0.

© 1987 Optical Society of America.

application program for performing tasks which require expertise. While there is no necessary connection between expert systems and abductive reasoning, most expert systems perform abductive tasks. Conversely, most of the standard examples of programs which do abductive reasoning in the presence of uncertainty are expert systems. With these considerations, we can refer to the proposed rule-based scheme as an expert system.

The definition of the problem is given in Sec. II, and the concept of dividing an image into partitions is explained there. The various considerations for correlation filter synthesis (i.e., their size, number, output assignments, and training sets) are discussed in Sec. III. A statistical motivation for the proposed scheme is advanced in Sec. IV along with illustrative examples. Section V is a description of the rule-based symbolic processor, and how expertise and evidence are incorporated into the program. Initial test results are reported in Sec. VI. A summary of the paper is given in Sec. VII.

II. Problem Definition

We wish to design a system capable of identifying, recognizing and classifying objects in the face of 3-D distortions. Our case study is confined to a tank and an armored personnel carrier (APC). However, the basic concept has far more generality. The filter is intended to achieve aspect-invariant distortion invariance. To provide this, we employ training images (of the target objects at several different aspect views) as detailed elsewhere.[1] We partition these input training images into several subimages, and synthesize correlation filters for each partition. The goal is to use correlation filters to generate a multibit multiple filter description (or symbolic code) for each object for distortion and shift-invariant symbolic object classification.

Once representative images of each object have been selected for training the correlation filters, these training set images are partitioned into M $k \times k$ pixel subimages or partitions. We assume an input object resolution of $d \times d$ pixels. Thus,

$$M \cdot k^2 = d^2. \tag{1}$$

We use the symbol ω_{ij} to denote the ith subimage of the jth training image. Therefore $1 \leq i \leq M$ and $1 \leq j \leq N$. The terms partition and subimage will be interchangeably used in this discussion.

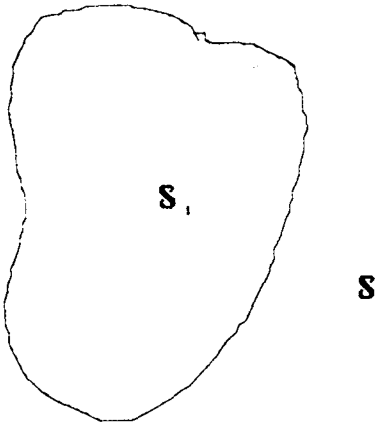


Fig. 1. Partitioning scheme for complete images.

Table I. Terms and Definitions for Filter Synthesis

Term	Definition	Value
d	1-D Image dimension	32
k	1-D Partition dimension	8
M	Number of partitions	16
N	Total number of training images	12

We propose that correlation filters $\mathbf{f}_i$ be synthesized for each partition, $1 \leq i \leq M$. The filters $\mathbf{f}_i$ are assumed to be functions of the training subimages ω_{ij} (for all j) and to be of dimensions $k \times k$. The correlation filter synthesis procedure is not important for the discussion in this paper. We use minimum average correlation energy (MACE)[6] filters in our work because of their time and memory efficient synthesis, and their ability to form good correlation peaks.

III. Criteria for Filter Synthesis

In this section, we discuss relevant synthesis criteria such as the designation of filter outputs and the selection of training sets for the filters. The proposed scheme is best described by means of the diagram in Fig. 1.

We use sixteen partitions ($M = 16$) in our work. The outputs from the corresponding sixteen filters are collectively denoted by the 16-element output vector $\mathbf{v}$. The layout is shown in Fig. 1. The image is divided into sixteen subimages, each of which is a partition. The partitions are numbered from 1 to 16 as in Fig. 1. The training set of the filter $\mathbf{f}_i$ $(1 \leq i \leq 16)$, corresponding to the ith partition is simply the collection of the ith subimages in all complete images in the data base. The training set for the ith filter is represented by $\phi_i = \{\omega_{ij}, j = 1, \ldots N\}$.

The data base chosen for our work consists of six complete images of the tank and six images of the APC. The images were taken at a depression angle of 60° and were evenly spaced every 60° about the normal. Since there were six images per class, the training set ϕ_i for each filter $\mathbf{f}_i$ included $2 \times 6 = 12$ subimages ω_{ij}, $1 \leq j \leq 24$. The data base images were 32×32 pixels (i.e., $d = 32$). Since $M = 16$, we select $k = 8$ to satisfy Eq. (1). The four synthesis parameter values for d, k, M, and N are listed in Table I. The entire data base contains seventy-two images, thirty-six of the tank and thirty-six of the APC, each image being a different aspect view with 10° increments in aspect angle used.

The desired filter outputs must also be specified for both classes of data. Two choices for the filter outputs are shown in Figs. 2(a) and (b). These were used for the tank and the APC, respectively. The value (1 or 0) in each square in the correlation output represents the output of the corresponding partition of the filter. Thus, as seen in Fig. 2(a), the sixteen filters $\mathbf{f}_i$ yield an output of 1 for odd values of i (and 0 otherwise), when the input image is a tank. This output vector for the

1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

a

0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

b

Fig. 2. (a) Partitioned output pattern for tank; (b) partitioned output pattern for an APC.

tank is denoted by $\mathbf{v}_1$, which is obtained by lexicographically ordering the elements of Fig. 2(a). Similarly, Fig. 2(b) shows the desired outputs for an APC input. The corresponding output vector is denoted by $\mathbf{v}_2$. If the filter outputs are set to be $\mathbf{v}_1$ or $\mathbf{v}_2$ for each target class for all images in the data base, the output vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are invariant to 3-D distortions of the targets to be classified. During filter synthesis, we specify that the training set objects have these two output patterns. Thus, we achieve a unique 16-bit symbolic correlation output description for each input object.

The Fourier transform of the filter $\mathbf{f}$ (with $M = 16$ outputs as shown in Fig. 2 in the space domain) is synthesized as a matched spatial filter in the frequency domain of a frequency plane correlator.[7] This produces one filter with each of its $M = 16$ partitions on a different spatial frequency carrier (with frequency proportional to the subimage's location in the filter). The correlation output for such a filter yields a 4 × 4 array of correlation values (a 16-bit symbol) for each occurrence of a tank or APC in the input. The symbolic pattern of Fig. 2(a) will result when the input is a tank and the pattern in Fig. 2(b) will result when the input is an APC. The spatial location of the pattern denotes the object's position in the input image plane. Thus, one uses such a correlator in the conventional manner but searches the correlation plane for specific 4 × 4 symbolic patterns, descriptive of different objects.

IV. Statistical Motivation

A statistical motivation for the proposed scheme may be gained from the following considerations. Typically, the pattern recognition schemes that use correlation filters have a high false alarm rate. The problem of false alarms has not been addressed fully and is an important topic for future research. In this paper, we briefly describe a potential solution to the problem, but will defer the details of the analysis to a future publication.

Consider a single correlation filter employed for target recognition. When the correct object is present at the input, the output correlation peak is at a user-specified value. This is true provided the filter is distortion invariant (one approach to this is to make the proper choice of the training set images). The value of the output peak determines the class of the input image. Unfortunately, it can be shown that an infinite number of images exist that yield correlation peak outputs exactly equal to those specified during filter synthesis by the user. Thus, even in the absence of any target, the filter may output correlation values equal to or close to those specified for targets and thereby give rise to false alarms. Decisions based on a single filter are hence unreliable. In formal terms, the constraints imposed during filter synthesis are necessary but not sufficient for target recognition.

It can be shown that the simultaneous use of more than one filter reduces the false alarm rate. The simultaneous use of multiple filters (such as the K-tuple SDF) has been suggested in previous research (although not for these specific reasons). Our present scheme based on partitioned images achieves a lower false alarm rate because more constraints have to be satisfied simultaneously. As stated earlier, we do not provide a detailed analysis in this paper. However, we now offer intuitive insight into the problem and its solution.

In the following, we shall represent a d-dimensional vector space S and its subsets S_i by plane figures as in Fig. 3. The plane S represents the whole set of possible images that could ever appear at the input of the correlator. Assume that a filter $\mathbf{f}_1$ is synthesized such that an output of u_1 is obtained whenever the target is present at the input. Since images other than the target exist that yield an output u_1, we denote the subspace of all such images by the region S_1. Thus all images inside S_1 are potential sources of false alarms with the filter $\mathbf{f}_1$. Now assume that we employ M filters $\mathbf{f}_i$, $1 < i < M$. For each filter $\mathbf{f}_i$, there exists a subspace of images S_i (similar to S_1) that yield false alarms. All images in S_i thus satisfy the constraints imposed on the filter $\mathbf{f}_i$ during synthesis. The M subspaces S_i for the M filters are shown in Fig. 4. By definition, all these subspaces must contain the training set images, and hence must have a nonzero intersection I. Moreover, an image must belong to this intersection to simultaneously satisfy all M filters. For a multifilter system, a false alarm is said to occur if, in the absence of a target, all M filters output correct correlation values. Therefore for a false alarm to occur with multiple filters, the input must yield M cor-

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Fig. 3. Domain S_1 of false alarm images in the space S of all images (for the case of a single filter).

rect outputs simultaneously. Only images in the intersection region have this property (since images in I by definition yield correct outputs for $\mathbf{f}_1,\mathbf{f}_2,\ldots,\mathbf{f}_M$). Thus images that cause false alarms with M filters must belong to the intersection set I. From Fig. 4 it is evident that the number of false alarms is less for M filters (than for any single $\mathbf{f}_i$) since the intersection I is smaller than any of the individual subspaces S_i. Moreover, the intersection becomes smaller as the number of filters (and hence the number of subspaces that must intersect) increases, indicating a diminishing false alarm rate for a larger number of image partitions.

The information in Fig. 4 can be interpreted in terms of the probability of false alarms. It can be shown, in rather general conditions, that a system using filters synthesized from complete images (without partitioning the data) has a higher probability of false alarm than a system employing multiple filters. The symbolic and associative postprocessing we perform allows flexibility in assigning objects to a class when the intersection region I in Fig. 4 becomes too small for a given set of data.

V. Probabilistic Rule-Based Recognition

In this section, we describe criteria for basic rule formulation for the recognition of targets using the output symbolic vectors $\mathbf{v}_n$. Guidelines are provided for incorporating new rules into the system, via interactive exchange of information. The criteria for assigning confidence measures (probabilities) to each decision are also discussed.

We wish to determine the conditional probability P (Tank/$|\mathbf{v}| > T$) (i.e., the probability that the input image is a tank given the observation $\mathbf{v}$ where T is a threshold value). A purely statistical solution to the problem would be to obtain estimates for $P(|\mathbf{v}| > T)$ and to use Bayes rule[8] to obtain an estimate for $P(\text{Tank}/|\mathbf{v}| > T)$, assuming *a priori* probabilities for $P(\text{Tank})$. However, it is generally difficult to obtain all the necessary estimates for $P(|\mathbf{v}| > T)$, because of the large number of possibilities. Thus we resort to abductive reasoning to provide a solution.

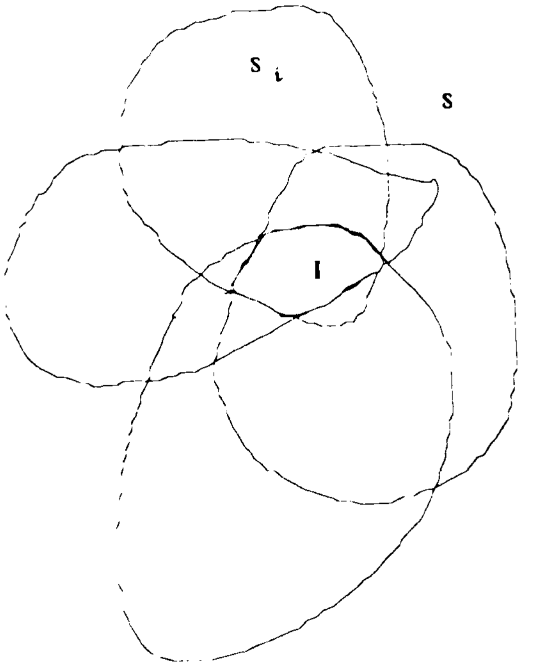


Fig. 4 Intersection I of multiple filter domains S_i for the case of multiple filters.

Given a measured output vector $\mathbf{v}$, the system determines a limited number of ways in which the observation could have resulted from image distortions, missing parts, etc., and the probability associated with each. The system then uses abductive reasoning to determine possible output filter element errors. Once a filter output is suspected of error, its symbolic value is altered to test for better matches with the descriptions stored in memory. During system test runs, we develop an *a priori* belief in specific filter outputs by observing that some filter outputs are in error less frequently than others. In operation the system is then instructed to examine these more reliable filter outputs in certain conditions and to ignore other symbolic outputs. The decisions made in such conditions (i.e., ignoring certain symbols) are assigned a lower confidence. We now detail these techniques.

A. Rule Formation Introduction

Target recognition is a trivial task if the input image is represented in the data base. In this case, the output vector is expected to exactly match the 16-bit patterns in Fig. 2(a) or (b). We will refer to the proper output vector ($\mathbf{v}_1$ or $\mathbf{v}_2$) simply as the output vector $\mathbf{v}$. A simple rule for target recognition in this case is:

Rule 1:

(1) Assign the symbol A to the symbolic outputs that are 1, and the symbol B to outputs that are 0.

(2) If elements (1,5,9,13) and (3,7,11,15) of $\mathbf{v}$ are A and elements (2,6,10,14) and (4,8,12,16) are B, the input is a tank with confidence = 1.0.

(3) Else, if the complement of 2 is true, the input is an APC with confidence = 1.0.

(4) Else, set error flag (1) and confidence = 0.0.

End rule 1.

This rule operates on the output vector $\mathbf{v}$. We treat the outputs 1 and 0 as symbolic values and assign them the symbols A and B. The complement rule[9] in step (3) evaluates the complement of the rule in step (2). If $\mathbf{v}$ does not satisfy the rule, this is a procedure error and the flag in step (4) is used to record this fact. Sec. V.C provides further rules and how they are learned.

B. Multiple Filter Banks

In a real environment, it is unlikely that input images will perfectly match any image in the data base, since input images can be distorted by 3-D rotations of the target or by occlusion of target parts by natural and man-made artifacts. Our processor adapts to such situations as we now describe.

To improve the decision making process, we employ a set of S symbolic filters (with M partitions in each).

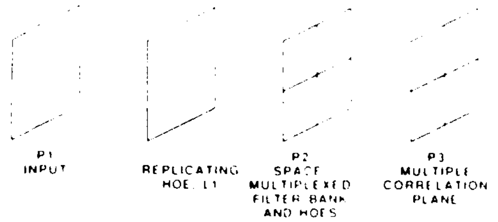


Fig. 5. Spatially multiplexed multiple filter bank correlator.

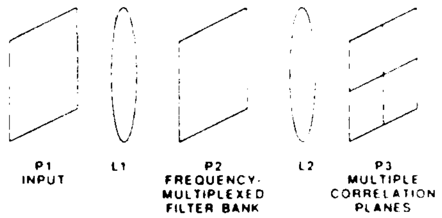


Fig. 6. Frequency-multiplexed multiple filter bank correlator.

We refer to the set of filters as a filter bank. We thus perform the correlation of the input data with S filters. For a single input object, there will be S output correlation planes and each will contain an M element output vector $\mathbf{v}_s$ (the symbolic pattern chosen).

Figure 5 shows a correlator with $S = 4$ multiple correlation planes at P_3 that are the correlation of the P_1 data with $S = 4$ different spatially multiplexed filters at P_2. The holographic optical element (HOE) $L1$ provides a spatial replication of the Fourier transform of the P_1 data at four separate locations in P_2. Four space-multiplexed filters with HOE Fourier transform lenses are used at P_2. One can also achieve multiple correlations using frequency-multiplexed filters at P_2 as shown in Fig. 6. In both architectures, each correlation plane contains a 4 × 4 spatial pattern (the symbolic code chosen, such as those in Fig. 2) at spatial locations corresponding to each occurrence of one of the objects in the P_1 data.

In our initial symbolic processor tests, we used $S = 3$ filter banks with $M = 16$ filters in each. Each object is thus described by three vectors $\mathbf{v}_{s1}$, $\mathbf{v}_{s2}$, $\mathbf{v}_{s3}$, with a total of 3 × 16 = 48 elements. Figure 6 shows the second output vectors $(\mathbf{v}_{s2},1)$ and $(\mathbf{v}_{s2},2)$ for the class 1 and 2 objects and the third output vectors $(\mathbf{v}_{s3},1)$ and $(\mathbf{v}_{s3},2)$ chosen. Each vector pair is a vector and its complement. The advantage of using a filter bank is that an error in one output vector can be confirmed (or invalidated) using the remaining $S - 1$ output vectors. We now describe how rules were developed interactively to achieve this.

C. Interactive Knowledge Acquisition

In each object class, thirty-six images (at 10° aspect increments) exist. The filters $\mathbf{f}_i$ for the various filter banks were formed from six images/class (at 60° increments in aspect), i.e., using twelve of the seventy-two possible images in the 2 classes (tank and APC). The three filter banks were formed and encoded as in Figs. 2 and 7. The three filter output vectors $\mathbf{v}_{s1}$ to $\mathbf{v}_{s3}$ obtained were measured and stored. Although the ideal symbolic patterns contain ones and zeros, the actual filter outputs are values between 0 and 1 (partial truth).

1	0	0	1
0	1	1	0
0	1	1	0
1	0	0	1

a

0	1	1	0
1	0	0	1
1	0	0	1
0	1	1	0

b

1	1	1	1
1	0	0	1
1	0	0	1
1	1	1	1

c

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

d

Fig. 7. Partitioned output patterns for (a) tanks and (b) APCs from filter 2, and for (c) tanks and (d) APCs from filter 3.

The program first attempts to classify the output vectors using rule 1 (Sec. V.A) applied to all three vectors $\mathbf{v}_{s1}$, $\mathbf{v}_{s2}$, and $\mathbf{v}_{s3}$. For a decision to be made, all three output vectors must satisfy rule 1. If a decision is not possible, it is assumed that errors have occurred in the vectors that failed rule 1. The user is interrogated for the class of the input image. The three output vectors and the user's choice for object class are stored. The program proceeds in this manner until this information has been obtained on all seventy-two images. After storing the three output vectors and the user-specified class for all test images, the program interrogates the user for the number of rules that should be used for decision making. An iterative search[10] is then initiated to find these rules, such that the number of errors obtained using each rule is as small as possible. We now detail this procedure.

To illustrate this procedure, consider the tank images as inputs. It is found that for thirty of the thirty-six tank images (i.e., all nontraining set images), the fourth element of vector $\mathbf{v}_{s1}$ is in error [i.e., it should be 0 as shown in Fig. 2(a), but was 1]. It is also found that for the same thirty test images, the seventh and eighth elements of $\mathbf{v}_{s2}$ and the twelfth element of $\mathbf{v}_{s3}$ are in error. Therefore a possible second rule is:

Rule 2: failing rule 1 then

(1) If all elements of $\mathbf{v}_{s1}$ match except for element (4) and $\mathbf{v}_{s2}$ matches except for elements (7,8) and $\mathbf{v}_{s3}$ matches except for element (12), the input is a tank

with probability = 0.86 (confidence = 0.79).
(2) Else: if the complement (of step 1) is true, the input is an APC with probability = 0.80 (confidence = 0.73).
(3) Else, set error flag (2) and confidence = 0.0.
End rule 2.

Using this rule, it was found that thirty-one out of the thirty-six tank images satisfied the match requirements, and hence were correctly identified (while none of the APC test images satisfied the rule). Thus the probability that an image that satisfies rule 2 is a tank is 31/36 = 0.86. This is how the probability values noted in steps (1) and (2) in rule 2 were obtained. If the match technique fails for the tank, the complementary rule is evaluated for APCs as in step (2). It was found that twenty-nine APCs satisfied the complementary rule, and thus the probability for APCs is estimated to be 29/36 ≈ 0.80 as noted in step (2).

It is necessary to distinguish between confidence and probability measures. As the number of the rule used increases, more and more vector elements are ignored. Since fewer symbols are taken into consideration, the confidence in higher rule must be lower. However, the probability that higher rules are satisfied is larger, because fewer vector elements are used for making a decision. Thus we need to compensate by including the number of elements examined in the expression for the confidence. This is easily done by setting

$$\text{confidence} = \text{probability} \times \frac{\text{number of elements examined}}{\text{total number of elements}} . \quad (2)$$

For rule 1, we use a confidence of 1.0, since if it is satisfied, we have perfect confidence (ignoring the possibility of false alarms) in the class estimate it gave. For rule 2, the confidence from (2) is 0.86(44/48) = 0.79 and 0.80(44/48) = 0.73 for the tanks and the APCs, respectively. Thus for low numbered rules (using more of the vector elements), the confidence is approximately equal to the probability that the rule is satisfied (since the number of symbols used for decision making is close to the total number of symbols). However, for higher numbered rules, the confidence is a fraction of the probability, reflecting the fact that some information was ignored in making the decision. We used five rules. The data for these are provided in Table II for rules 1–5 and their complements 1c–5c. The confidence of each rule decreases as expected and the number of symbolic elements (out of forty-eight) ignored increases as shown.

The procedure failing rule noted at the start of rule 2 checks the error table to see if a given rule was violated by the output vectors. This is required for determining branch and termination conditions and is particularly useful in programs with intricate feedback routes. Since our rules have a precedence hierarchy, the procedure failing rule is not absolutely necessary for our present program execution. However, we included it to accommodate the future development of the program into a more complex rule-based algorithm. Note that if any one of the S output vectors does not satisfy a particular rule, the condition for failure is set.

Table II. Confidence Values for a Five-Rule System

Rule number	Class	Confidence	Number of elements ignored out of 48
1	Tank	1	0
2	Tank	0.79	4
3	Tank	0.76	9
4	Tank	0.62	13
5	Tank	0.48	17
1c	APC	1	0
2c	APC	0.73	4
3c	APC	0.69	9
4c	APC	0.63	13
5c	APC	0.47	17

Using rule 1 before rule 2 establishes a hierarchy for rule usage. If an image is found to satisfy rule 1, it is easily classified as either a tank or an APC. However, most images may not satisfy rule 1, and rule 2 must be applied as a second test. This rule is estimated to be correct 86% of the time for tanks and 80% of the time for the APCs. This percentage probability that the rule is satisfied is then used in Eq. (2) to obtain a confidence measure. All images satisfying rule 1 will satisfy rule 2 also. The purpose of the interactive procedure leading to our five rules is to determine the most reliable symbols and the probabilities and confidence of the class estimates for each rule. This general technique to obtain the rules to be used results in a final set of rules that is a decision tree. The technique used to select the symbols used at successive levels is general and can be applied to many problems. It is not domain specific. The specific rules that result will differ for each data set. Thus the method adapts to different knowledge sources.

We now discuss rules that use the information in one output vector to rectify errors in the others using a new symbolic substitution rule. For example, suppose that the fourth element of the vector $\mathbf{v}_{s_1}$ is in error for a particular input image. The program assumes that the part of the image in the fourth partition is missing, or is severely distorted. Therefore, it assumes that the fourth elements of vectors $\mathbf{v}_{s_2}$ and $\mathbf{v}_{s_3}$ are also in error (since the replicas of the same image are input to all filter banks). This rule module then alters the fourth elements of $\mathbf{v}_{s_2}$ and $\mathbf{v}_{s_3}$ and checks to see if use of the original or altered $\mathbf{v}_{s_2}$ and $\mathbf{v}_{s_3}$ vectors yields a better match. Both possibilities are considered, since if the proper element value is 0, it may not be altered, whereas if its proper value were 1, it may be altered; or vice versa, depending on the nature of the difference in the corresponding region of the input. If altering the output vectors in this manner provides a better match, the assumption that a part of the image is distorted or missing is validated. The input image is then classified appropriately. In principle, this symbolic substitution can be applied to more than one element of the output vectors. This rule module would be applied to each rule and then (if no match is obtained) the next rule would be accessed. A straightforward procedure can be devised to identify which elements of the output vectors may be in error. A major advantage of a rule-based recognition technique is the ability to anticipate and correct errors before a decision is made.[10]

The rules (see Table II) with highest confidence are invoked first, since a later rule provides a lower confidence than the previous rule. We emphasize that the process of rule generation is an off-line interaction between the programmer and the computer. Once the set of rules is formulated, the program stores them in the memory for on-line access. The technique used for generating the rules attempts to maintain a hierarchy, such that images that satisfy rules with higher confidence will also satisfy rules with lower confidence. This occurs in the present case. In general, most images will satisfy more than one rule. The decision with the highest vote of confidence is accepted as the best choice for image classification.

D. Associative Memory

If the confidence of the lowest rule with a match is felt to be too small, the rule-based decision making is deemed to be unreliable. In our five-rule system, we always have a confidence of at least 0.48. However, this will not be sufficient in most cases. Use of rules beyond rule 3, where the confidence drops below 70%, will generally not result in acceptable performance. In such situations, the program resorts to matching the $\mathbf{v}_s$ vectors to the closest ideal output set of vectors (by minimizing the norm of the difference between the two vectors). This is analogous to the information retrieval process in an associative memory. Thus, we call an associative processor one that returns three vectors closest to the computed vectors $\mathbf{v}_{s1}$, $\mathbf{v}_{s2}$, and $\mathbf{v}_{s3}$. We could use S separate associative processors (one for each of the S output vectors) to reduce crosstalk or interference between symbols. With only sixteen element vectors, there is considerable crosstalk. At present, we employ one autoassociative processor that handles all six vectors (three per class object).

The design of associative memory processors is discussed elsewhere[11] and is not reviewed here. The autoassociative memory matrix is given by the Moore-Penrose generalized inverse

$$\mathbf{M} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T, \tag{3}$$

where the six columns of the matrix $\mathbf{X}$ are the three $\mathbf{v}_{sn}$ vectors for each class. The output vector that results will be a minimum mean-square approximation to the ideal data. While most errors in the input vector are corrected by such associative processing, a few correct symbol values may be altered (i.e., errors can be introduced by the associative processor) to achieve the minimum error value. The error correcting capability of the associative processor depends on the size of the vector space, and the number of vectors stored, and is better for higher dimensional input vectors. In our case, the dimensionality of one input vector is 16, which is relatively low. Thus dramatic improvements are not expected. We could employ the three vectors as one 48-element input vector and thereby improve the performance of the associative processor. However, our present purpose is not for the associative processor to fully correct the input vector, but for the combination of an associative processor and our rule-based symbolic processor to be used. Memory size and performance studies will determine the best symbolic vector dimensionality to be employed. The output vector obtained from the associative processor we used is thresholded at 0.5 to obtain binary valued symbolic vector elements. These resulting vectors are then fed to our rule-based processor, which is then checked for an improvement in the confidence of the class estimate. An improvement is not always guaranteed since the associative processor can change correct symbols also as noted at the outset.

VI. Initial Test Results

We now discuss the initial performance of our rule-based symbolic processor. A bank of three filters was formed with symbolic outputs for 2 classes as shown in Figs. 2 and 7 from six images per class of aspect-distorted tanks and APCs. A set of five rules for our rule-based system was produced. Rules 1 and 2 were presented earlier in Secs. V.A and V.C. Subsequent rules were obtained similarly by noting which symbolic elements were generally in error. Table II summarizes the confidence for each rule for each object class. The confidence is obtained as detailed in Sec. V.C and it is seen to decrease for subsequent rules. This is expected since, with fewer symbols used in subsequent rules, we expect lower confidences in the class estimates produced. The rules were then applied to the training set images, and 100% correct results were obtained (with confidence 1.0) as expected (see Table III). This confirmed the proper synthesis of the symbolic filters.

The system was then tested with five images per class (only one of these images per class was a training image, the 0° view) with partitions 7 and 10 (see Fig. 1) of each image removed to simulate data occlusion and to test the system's performance. Table IV shows the results. The first three columns in Table IV give the test number, the aspect view, and the type of object. The last three columns show the results obtained: the class estimate, the rule number which the object first passed, and the confidence of the rule (and hence the confidence of the class estimate). As seen, one error was obtained (for the class estimate in test 4). The confidence of this estimate is low (62%) and thus would be suspect. The remaining objects are correctly classified with a confidence of at least 69%.

The error case in Table IV would be sensed by its low confidence and thus the associative memory would be used. For this case, the three distorted output vectors $\mathbf{v}_{s1}$, $\mathbf{v}_{s2}$, and $\mathbf{v}_{s3}$ computed for this image were fed to an autoassociative processor whose memory contained the ideal vector patterns. The output obtained from the associative processor for each $\mathbf{v}_{sn}$ input is a linear combination of the ideal stored vectors. This output was thresholded at 0.5 to obtain three new output vectors. Our rule-based system was then again applied to these new vectors. The resulting decision in this case was correct (i.e., the image in test 4 was now identified as a tank) using rule 3 with a confidence of

Table III. Results of Tests Using the Twelve Training Set Tank and APC Images

Test number	Aspect view (deg)	Actual class	Class estimate	Rule number	Confidence
1	0	Tank	Tank	1	1.0
2	60	Tank	Tank	1	1.0
3	120	Tank	Tank	1	1.0
4	180	Tank	Tank	1	1.0
5	240	Tank	Tank	1	1.0
6	300	Tank	Tank	1	1.0
7	0	APC	APC	1	1.0
8	60	APC	APC	1	1.0
9	120	APC	APC	1	1.0
10	180	APC	APC	1	1.0
11	240	APC	APC	1	1.0
12	300	APC	APC	1	1.0

Table IV. Results of Tests Using Five Tank and Five APC Test Set Images with Two of the Sixteen Partitions of Each Image Omitted (by Occlusion)

Test number	Aspect view (deg)	Actual class	Class estimate	Rule number	Confidence
1	0	Tank	Tank	1	1.0
2	20	Tank	Tank	1	0.79
3	50	Tank	Tank	1	0.79
4	90	Tank	APC	1	0.62
5	110	Tank	Tank	1	0.76
6	0	APC	APC	1	1.0
7	20	APC	APC	1	0.69
8	50	APC	APC	1	0.80
9	90	APC	APC	1	0.69
10	110	APC	APC	1	0.69

0.76. Examining the input and output vectors from the associative memory, we found that thirteen symbols were in error prior to associative processing, and that the number of symbol errors was reduced to nine by the associative processor. In this case, none of the symbols that were originally correct was found to be in error after associative processing.

We have thus seen an example when an associative processor can be used to correct errors in the output vectors v_i. This occurs because the associative processor effectively utilizes all available information to make an optimal mathematical guess. Unlike an associative processor, the proposed rule-based processor only examines the most reliable symbols, and hence ignores some information at each rule. The flexibility of the rule-based processor is in its ability to provide a logical decision (along with a confidence measure) even when the input information is incomplete. This was successfully demonstrated in our initial tests in Table IV. We expect that the use of autoassociative memories in a rule-based symbolic processor will improve performance of the system in most cases (as long as the number of errors is modest, the number of classes is not excessive, and the dimensionality of the symbolic vectors is sufficiently large).

VII. Conclusion

In this paper, we have outlined a system capable of recognizing targets even when parts of the object are not visible. Motivation was provided for filtering by parts and an example was given to illustrate the possible advantages of synthesizing symbolic correlation filters formed from subimages of objects. A system was devised and simulated for demonstration purposes. Initial simulation results were encouraging and demonstrated 3-D distorted object recognition with occluded object parts. We also showed that an associative processor can be used in conjunction with the rule-based system to improve performance. We detailed how the system's rules are developed via off-line interactions between the programmer and the computer. The use of symbolic substitution for error compensation was also suggested.

Further tests with this concept and its various aspects are required. This requires devising more robust rules. It also includes further use of the ability of the system to predict errors and compensate for them using multiple filter banks. The use of abductive reasoning for developing the programs necessary for this appears quite attractive. Efficient methods of updating the correlation filters on-line (involving the addition or removal of training images) and the memory storage requirements of such a system are other topics for future investigations.

We acknowledge the support of different aspects of this work by various contractors (General Dynamics Valley Systems Division, the Defense Advanced Research Projects Agency, and the Air Force Office of Scientific Research).

References

1. D. Casasent and W. T. Chang, "Correlation Synthetic Discriminant Functions," Appl. Opt. 25, 2343 (1986).
2. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation," Appl. Opt. 23, 1620 (1984).
3. Y. Shirai, K. Koshikawa, M. Oshima, and K. Ikeuchi, "Applications of 3-D Models to Computer Vision," Comput. Graphics 1, 269 (1983).
4. R. Ducan, "Three-Space Hidden Surface Removal Using Boundary Traversal Logic," Comput. Aided Des. 15, [illegible]
5. E. Charniak and D. McDermott, *Introduction to Artificial Intelligence* (Addison-Wesley, Reading, MA, 1985), pp. 21 and [illegible]
6. A. Mahalanobis, B. V. K. Vijaya Kumar, and D. Casasent, "Minimum Average Correlation Energy Filters," Appl. Opt. 26, 3633 (1987).
7. A. B. VanderLugt, "Signal Detection by Complex Matched Spatial Filtering," IEEE Trans. Inf. Theory IT-10, 139 (1964).
8. A. Papoulis, *Probability, Random Variables, and Stochastic Processes* (McGraw-Hill, New York, 1984).
9. K. H. Brenner and A. Huang, "An Optical Processor Based on Symbolic Substitution," in *Technical Digest of Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1985), paper WA4.
10. P. H. Winston, *Artificial Intelligence* (Addison-Wesley, Reading, MA, 1979), pp. 179-200.
11. T. Kohonen, *Self-Organization and Associative Memory* (Springer-Verlag, New York, 1984).

12. PUBLICATIONS, PRESENTATIONS AND THESES PRODUCED

Publications from the start of this grant are listed in Section 12.1. Papers published previously are noted in Section 12.1.1. Papers published during the most recent year (1987) of this grant are listed in Section 12.1.2. Books and book chapters published are noted in Section 12.1.3. Presentations given during the duration of this grants are noted in Section 12.2. Theses that were supported by this AFOSR research are noted in Section 12.3. The wealth of documentation provided under this AFOSR grant is quite phenomenal. This includes over 90 papers and over 100 presentations in diverse journals and communities.

12.1 PUBLICATIONS (AFOSR SUPPORTED, SEPTEMBER 1984-DATE)

12.1.1 PAPERS PUBLISHED UNDER AFOSR SUPPORT (SEPTEMBER 1984-DECEMBER 1986)

1. D. Casasent and R.L. Cheatham, "Hierarchical Pattern Recognition Using Parallel Feature Extraction", *Proc. ASME, Computers in Engineering 1984*, Vol. 1, pp. 1-6, August 1984.

2. D. Casasent, A. Ghosh and C.P. Neuman, "Iterative Solutions to Nonlinear Matrix Equations Using a Fixed Number of Steps", *Proc. SPIE*, Vol. 495, pp. 102-108, August 1984.

3. R.L. Cheatham and D. Casasent, "Hierarchical Fisher and Moment-Based Pattern Recognition", *Proc. SPIE*, Vol. 504, pp. 19-26, August 1984.

4. W.T. Chang, D. Casasent and D. Fetterly, "SDF Control of Correlation Plane Structure for 3-D Object Representation and Recognition", *Proc. SPIE*, Vol. 507, pp. 9-18, August 1984.

5. D. Casasent, A. Goutzoulis and B.V.K. Vijaya Kumar, "Time-Integrating Acousto-Optic Correlator: Error Source Modeling", *Applied Optics*, Vol. 23, pp. 3230-3237, September 1984.

6. D. Casasent and R.L. Cheatham, "Image Segmentation and Real-Image Tests for an Optical Moment-Based Feature Extractor", *Optics Communications*, Vol. 51, pp. 227-230, September 1984.

7. D. Casasent and V. Sharma, "Feature Extractors for Distortion-Invariant Robot Vision", *Optical Engineering*, Vol. 23, pp. 492-498, September/October 1984.

8. V. Sharma and D. Casasent, "Optimal Linear Discriminant Functions", *Proc. SPIE*, Vol. 519, pp. 50-55, October 1984.

9. D. Casasent, W. Rozzi and D. Fetterly, "Projection Synthetic Discriminant Function Performance", *Optical Engineering*, Vol. 23, pp. 716-720, November 1984.

10. W.T. Chang and D. Casasent, "Chord Distributions in Pattern Recognition: Distortion-Invariance and Parameter Estimation", *Proc. SPIE*, Vol. 521, pp. 2-6, November 1984.

11. A. Goutzoulis, D. Casasent and B.V.K. Vijaya Kumar, "Acousto-Optic Processor for Adaptive Radar Noise Environment Characterization", *Applied Optics*, Vol. 23, pp. 4303-4308, December 1984.

12. D. Casasent, "Optical Processing Research Making Significant Advancements", *Laser Focus*, pp. 150, October 1984.

13. D. Casasent, "Coherent Optical Pattern Recognition: A Review", *Optical Engineering*, Vol. 24, Special Issue, pp. 26-32, January 1985.

14. D. Casasent and J.Z. Song, "A Computer Generated Hologram for Diffraction-Pattern Sampling", *Proc. SPIE*, Vol. 523, January 1985.

15. D. Casasent, "Hybrid Optical/Digital Image Pattern Recognition: A Review", *Proc. SPIE*, Vol. 528, pp. 64-82, January 1985.

16. D. Casasent, "Computer Generated Holograms in Pattern Recognition: A Review", *Proc. SPIE*, Vol. 532, pp. 106-118, January 1985.

17. D. Casasent, "Parallel Coherent Optical Processor Architectures and Algorithms for ATR", *Proc. of the Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition*, Leesburg, Virginia, July 1984, Published February 1985, pp. 33-49.

18. D. Casasent, "Frequency-Multiplexed Acousto-Optic Architectures and Applications", *Applied Optics*, Vol. 24, March 1985.

19. D. Casasent, "Fabrication and Testing of a Space and Frequency-Multiplexed Optical Linear Algebra Processor", *OSA Topical Meeting on Optical Computing*, pp. TuD7-1 - TuD7-4, March 1985.

20. D. Casasent and R.L. Cheatham, "Hierarchical Feature-Based Object Identification", *OSA Topical Meeting on Machine Vision*, pp. ThD4-1 - ThD4-4, March 1985.

21. D. Casasent and A. Mahalanobis, "Correlation Filters for Distortion-Invariance and Discrimination", *OSA Topical Meeting on Machine Vision*, pp. FB5-1 - FB5-3, March 1985.

22. D. Casasent and B.K. Taylor, "Banded-Matrix High-Performance Algorithm and Architecture", *Applied Optics*, Vol. 24, pp. 1476-1480, 15 May 1985.

23. D. Casasent and A. Ghosh, "Optical Linear Algebra Processors: Noise and Error-Source Modeling", *Optics Letters*, Vol. 10, pp. 252-254, June 1985.

24. D. Casasent, "Computer Generated Holograms in Pattern Recognition: A Review", *Optical Engineering*, Vol. 24, pp. 724-730, September/October 1985.

25. B.V.K. Vijaya Kumar and C. Carroll, "Loss of optimality in cross-correlators", *JOSA-A*, Vol. 1, pp. 392-97, 1984.

26. B.V.K. Vijaya Kumar, "Lower bound for the suboptimality of cross-correlators", *Applied Optics*, Vol. 23, pp. 2048-49, 1984.

27. B.V.K. Vijaya Kumar and C. Carroll, "Performance of Wigner distribution function based detection methods", *Optical Engineering*, Vol. 23, pp. 732-37, 1984.

28. B.V.K. Vijaya Kumar and C. Carroll, "Effects of sampling on signal detection using the cross Wigner distribution function", *Applied Optics*, Vol. 23, pp. 4090-94, 1984.

29. B.V.K. Vijaya Kumar, "Effect of signal bandwidth on the accuracy of signal reconstruction from its phase", *IEEE Trans. ASSP*, Vol. 32, pp. 1238-39, 1984.

30. B.V.K. Vijaya Kumar and C. Rahenkamp, "An optical/digital hybrid system for calculating geometric moments", *SPIE Proc.*, Vol. 579, pp. 215-224, 1985.

31. B.V.K. Vijaya Kumar, "Optimality of projection synthetic discriminant functions", *SPIE Proc.*, Vol. 579, pp. 86-95, 1985.

32. B.V.K. Vijaya Kumar, C.P. Neuman and K. DeVos, "Wigner distribution applications for system identification", *Fourth ASSP Workshop on Multidimensional Signal Processing*, Leesburg, Virginia, October 1985.

33. A. Goutzoulis, D. Casasent and B.V.K. Vijaya Kumar, "Detector effects on time integrating correlators", *Applied Optics*, Vol. 24, pp. 1224-33, 1985.

34. A. Goutzoulis and B.V.K. Vijaya Kumar, "Detector size effects on peak-to-sidelobe ratio in bulk Acousto-Optic spectrum analyzer", *Optical Engineering*, Vol. 24, pp. 908-12, 1985.

35. D. Casasent, W. Rozzi and D. Fetterly "Correlation Synthetic Discriminant Functions for Object Recognition and Classification in High Clutter", *Proc. SPIE*, Vol. 575, August 1985.

36. D. Casasent and W.T. Chang, "Parameter Estimation and In-Plane Distortion Invariant Chord Processing", *Proc. SPIE*, Vol. 579, pp. 2-10, September 1985.

37. D. Casasent, "Optical Processing Techniques for Advanced Intelligent Robots and Computer Vision", *Proc. SPIE*, Vol. 579, pp. 208-214, September 1985.

38. D. Casasent and H. Okuyama, "A High-Dimensionality Pattern Recognition Feature Space", *Proc. SPIE*, Vol. 579, pp. 245-257, September 1985.

39. W. Rozzi and D. Casasent, "Frequency-Domain Synthesis of Modified MSFs", *Optics Letters*, Vol. 10, pp. 517-519, November 1985.

40. D. Casasent, "Optical Computer Architectures for Pattern Analysis", *IEEE Computer Society Workshop* on Computer Architectures for Pattern Analysis and Image Database Management, Miami Beach, Florida, November 1985, pp. 49-52. IEEE Catalog No. 85CH2229-3. ISBN 0-8186-0662-2.

41. B.V.K. Vijaya Kumar and C. Rahenkamp, "Performance of a hybrid processor to compute geometric moments", *SPIE Proc.*, Vol. 638, pp. 32-40, March-April 1986.

42. B.V.K. Vijaya Kumar, "Geometric moments from Hartley transform intensities", *SPIE Proc.*, Vol. 639, pp. 253-59, March-April 1986.

43. B. Montgomery and B.V.K. Vijaya Kumar, "Nearest-neighbor non-iterative error correcting optical associative memory processor", *SPIE Proc.*, Vol. 638, pp. 83-90, March-April 1986.

44. B.V.K. Vijaya Kumar and E. Pochapsky, "Signal-to-noise ratio considerations in modified matched spatial filters", *JOSA-A*, Vol. 3, pp. 777-786, June 1986.

45. B.V.K. Vijaya Kumar and C. Rahenkamp, "Calculation of geometric moments using Fourier plane intensities", *Applied Optics*, Vol. 25, pp. 997-1007, 1986.

46. B.V.K. Vijaya Kumar and S. Rajan, "Subpixel delay estimation using group-delay functions", *SPIE Proc.*, Vol. 697, pp. 187-196, August 1986.

47. B. Montgomery and B.V.K. Vijaya Kumar, "Evaluation of the use of the Hopfield neural net model as a nearest-neighbor algorithm", *Applied Optics*, Vol. 25, pp. 3759-66, 15 October 1986.

48. B.V.K. Vijaya Kumar, "Minimum variance synthetic discriminant functions", *JOSA-A*, Vol. 3, pp. 1579-84, October 1986.

49. B.V.K. Vijaya Kumar, "Geometric moments computed from the Hartley transform", *Optical Engineering*, Vol. 25, pp. 1327-32, December 1986.

50. C. Rahenkamp and B.V.K. Vijaya Kumar, "Modifications to the McClellan, Parks

and Rabiner computer program for designing higher order differentiating FIR filters", *IEEE Trans. ASSP*, Vol. 34, pp. 1671-74, December 1986.

51. D. Casasent and W. Rozzi, "Modified MSF Synthesis by Fisher and Mean-Square Error Techniques", *Applied Optics*, Vol. 25, pp. 184-187, 15 January 1986.

52. D. Casasent and A.J. Lee, "A Feature Space Rule-Based Optical Relational Graph Processor", *Proc. SPIE*, Vol. 625, pp. 234-243, January 1986.

53. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations", *Proc. SPIE*, Vol. 625, pp. 220-225, January 1986.

54. D. Casasent, S.F. Xia, J.Z. Song, and A.J. Lee, "Diffraction Pattern Sampling Using a Computer-Generated Hologram", *Applied Optics*, Vol. 25, pp. 983-989, 15 March 1986.

55. D. Casasent, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence", *SPIE, Advanced Institute Series on Hybrid and Optical Computers*, Vol. 634, Leesburg, Virginia, March 1986.

56. D. Casasent and S.A. Liebowitz, "Model-Based System for On-Line Affine Image Transformations", *Proc. SPIE*, Vol. 638, pp. 66-75, March-April 1986.

57. D. Casasent, "Optical Computing at Carnegie-Mellon University", *Optics News, Special Issue on Optical Computing*, Vol. 12, pp. 11-13, April 1986.

58. D. Casasent and S.F. Xia, "Phase Correction of Light Modulators", *Optics Letters*, Vol. 11, pp. 398-400, June 1986.

59. D. Casasent, "Optical Artificial Intelligence Processors", *IOCC-1986 International Optical Computing Conference, Proc. SPIE*, Vol. 700, July 1986, pp. 246-250, 1986.

60. D. Casasent, J. Jackson and G. Vaerewyck, "Optical Array Processor: Laboratory Results", *IOCC-1986 International Optical Computing Conference, Proc. SPIE*, Vol. 700, July 1986, pp. 323-327, 1986.

61. D. Casasent and W.T. Chang, "Correlation Synthetic Discriminant Functions", *Applied Optics*, Vol. 25, pp. 2343-2350, 15 July 1986.

62. D. Casasent and B. Telfer, "Distortion-Invariant Associative Memories and Processors", *Proc. SPIE*, Vol. 697, August 1986.

63. D. Casasent and A.J. Lee, "An Optical Relational-Graph Rule-Based Processor for Structural-Attribute Knowledge Bases", *Applied Optics*, Vol. 15, pp. 3065-3070, 15 September 1986.

64. S.A. Liebowitz and D. Casasent, "Hierarchical Processor and Matched Filters for Range Image Processing", *Proc. SPIE, Vol. 727*, October 1986.

65. A. Mahalanobis and D. Casasent, "Large Class Iconic Pattern Recognition: An OCR Case Study", *Proc. SPIE*, Vol. 726, October 1986.

66. D. Casasent and W. Rozzi, "Computer-Generated and Phase-Only Synthetic Discriminant Function Filters", *Applied Optics*, Vol. 25, pp. 3767-3772, 15 October 1986.

67. D. Casasent, "Advanced Optical Pattern Recognition and Artificial Intelligence", *Laser Institute of America, ICALEO,* November 1986.

68. D. Casasent, "Optical Data Processing", Article in *Accent on Research*, p. 23, 1986 (Carnegie Mellon University Magazine).

69. A. Mahalanobis, B.V.K. Vijaya Kumar and D. Casasent, "Spatial-Temporal Correlation Filter for In-Plane Distortion Invariance", *Applied Optics,* Vol. 25, pp. 4466-4472, 1 December 1986.

70. J. Fisher, D. Casasent and C.P. Neuman, "Factorized Extended Kalman Filter for Optical Processing", Applied Optics, Vol. 25, pp. 1615-1621, 15 May 1986.

12.1.2 PAPERS PUBLISHED UNDER AFOSR SUPPORT IN THE PRESENT TIME PERIOD (JANUARY-DECEMBER 1987)

71. D. Casasent and R. Krishnapuram, "Detection of Target Trajectories Using the Hough Transform", *Applied Optics*, Vol. 26, pp. 247-251, 15 January 1987.

72. E. Baranoski and D. Casasent, "A Directed Graph Optical Processor", *Proc. SPIE*, Vol. 752, pp. 58-71, January 1987.

73. D. Casasent, A. Mahalanobis and S.A. Liebowitz, "Parameter Selection for Iconic and Symbolic Pattern Recognition Filters", *Proc. SPIE*, Vol. 754, pp. 284-303, January 1987.

74. D. Casasent and J.H. Song, "1-D Acousto Optic Processing of 2-D Image Data", *Proc. SPIE*, Vol. 754, pp. 64-73, January 1987.

75. D. Casasent, "Optical Pattern Recognition and Artificial Intelligence: A Review", *Proc. SPIE*, Vol. 754, pp. 2-11, January 1987.

76. D. Casasent, "Optical Pattern Recognition and AI Algorithms and Architectures for ATR and Computer Vision", *Proc. SPIE*, Vol. 755, pp. 83-93, January 1987.

77. D. Casasent, "Electro Optic Target Detection and Object Recognition", *Proc. SPIE*, Vol. 762, pp. 104-125, January 1987.

78. A.J. Lee and D. Casasent, "Computer Generated Hologram Recording Using a Laser Printer", *Applied Optics*, Vol. 26, pp. 136-138, 1 January 1987.

79. D. Casasent and E. Botha, "Knowledge in Optical Symbolic Pattern Recognition Processors", *Optical Engineering, Special Issue on Optical Computing and Nonlinear Optical Signal Processing*, Vol. 26, pp. 34-40, January 1987.

80. D. Casasent, "Optical Information Processing", *Sixth Edition, Encyclopedia of Science and Technology*, McGraw-Hill.

81. D. Casasent and R. Krishnapuram, "Curved Object Location by Hough Transformations and Inversions", *Pattern Recognition*, Vol. 20, No. 2, pp. 181-188, 1987.

82. D. Casasent, S.F. Xia, A.J. Lee and J.Z. Song, "Real-Time Deformation Invariant Optical Pattern Recognition Using Coordinate Transformations", *Applied Optics*, Vol. 26, pp. 938-942, 15 March 1987.

83. S.A. Liebowitz and D. Casasent, "Error Correction Coding in an Associative Processor", Applied Optics, Vol. 26, pp. 999-1006, 15 March 1987.

84. D. Casasent and A. Mahalanobis, "Rule-Based, Probabilistic, Symbolic Target Classification by Object Segmentation", *OSA Topical Meeting on Optical Computing* (March 1987), Technical Digest Series 1987, Vol. 11 (Optical Society of America, Washington, D.C., 1987), pp. 155-158.

85. D. Casasent and S.A. Liebowitz, "Model-Based Knowledge-Based Optical Processors", Applied Optics, Vol. 26, pp. 1935-1942, 15 May 1987.

86. R. Krishnapuram and D. Casasent, "Hough Space Transformations for Discrimination and Distortion Estimation", *Computer Vision, Graphics, and Image Processing*, Vol. 38, pp. 299-316, February 1987.

87. D. Casasent and A. Mahalanobis, "Optical Iconic Filters for Large Class Recognition", Applied Optics, Vol. 26, pp. 2266-2273, 1 June 1987.

88. R. Krishnapuram and D. Casasent, "Optical Associative Processor for General Linear Transformation", Applied Optics, Vol. 26, pp. 3641-3648, 1 September 1987.

89. A. Mahalanobis, B.V.K. Vijaya Kumar and D. Casasent, "Minimum Average Correlation Energy (MACE) Filters", Applied Optics, Vol. 26, pp. 3633-3640, 1 September 1987.

90. D. Casasent and B. Telfer, "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results", Proc. SPIE, Vol. 848, November 1987.

91. D. Casasent and S.I. Chien, "Rule-Based String Code Processor", Proc. SPIE, Vol. 848, November 1987.

92. D. Casasent and A. Mahalanobis, "Rule-Based Symbolic Processor for Object Recognition", Applied Optics, Vol. 26, pp. 4795-4802, 15 November 1987.

12.1.3 BOOK EDITING AND BOOK CHAPTERS

1. Intelligent Robots and Computer Vision, Ed. D. Casasent, SPIE, Vol. 726, October 1986.

2. Hybrid Image Processing, Ed. D. Casasent and A. Tescher, SPIE, Vol. 638, April 1986.

3. "Optical Feature Extraction", D. Casasent, Chapter in Optical Signal Processing, pp. 75-95, Ed. by J.L. Horner, Pub. by Academic Press, San Diego, 1987.

4. "Optical Linear Algebra Processors", D. Casasent and B.V.K. Vijaya Kumar, Chapter in Optical Signal Processing, pp. 389-407, Ed. by J.L. Horner, Pub. by Academic Press, San Diego, 1987.

12.2 PRESENTATIONS GIVEN ON AFOSR RESEARCH (AUGUST 1984-DATE)

September 1984

1. Philips Research Laboratories - Briarcliff, NY - "Optics and Pattern Recognition in Robotics".
2. Optical Society of America - Pittsburgh, PA, "CMU Center for Excellence in Optical Data Processing".
3. Carnegie-Mellon University, ECE Graduate Seminar - Pittsburgh, PA, "Optical Processing Research in the Center for Excellence in Optical Data Processing".
4. Westinghouse Corporation - Baltimore, MD, "Research and Facilities in the Center for Excellence in Optical Data Processing".

October 1984

5. Washington, D.C., "Optical Pattern Recognition: Feature Extraction".
6. Washington, D.C., "Optical Pattern Recognition: Correlators".
7. Washington, D.C., "Synthetic Discriminant Function Case Studies".
8. Washington, D.C., "Basic Optical Signal Processing Architectures and Algorithms".
9. Washington, D.C., "Advanced Optical Signal Processing Architectures and Algorithms".
10. Washington, D.C., "Optical Linear Algebra Processor Algorithms and Architectures".

11. Washington, D.C., "Optical Linear Algebra Processor Applications and High-Accuracy Architectures".
12. Carnegie-Mellon University, ECE Sophomore Seminar - Pittsburgh, Pennsylvania, "Research in the Center for Excellence in Optical Data Processing".
13. University of Pittsburgh, Center for Multivariate Analysis - Pittsburgh, PA, "Advanced Multi-Class Distortion-Invariant Pattern Recognition".
14. Wright Patterson Air Force Base - Ohio, "Multi-Functional Optical Signal Processor for Electronic Warfare".
15. George Mason University - Washington, D.C., "Optical Information Processing".
16. SPIE (IOCC) Conference - Boston, Massachusetts, "Optimal Linear Discriminant Functions".

November 1984

17. SPIE Robotics Conference - Boston, MA, "Chord Distributions in Pattern Recognition".
18. University of Maryland - "Optical Processing for Autonomous Land Vehicle Navigation".

January 1985

19. Fairchild Weston - Long Island, NY, "Optical Pattern Recognition and Optical Processing".
20. SPIE Conference - Los Angeles, CA, "Hybrid Optical/Digital Image Pattern Recognition: A Review".
21. SPIE Conference - Los Angeles, CA, "A Computer Generated Hologram for Diffraction-Pattern Sampling".
22. SPIE Conference - Los Angeles, CA, "A Recent Review of Holography in Coherent Optical Pattern Recognition".
23. Sandia National Laboratories - Albuquerque, NM, "Optical Pattern Recognition and Optical Processing".

February 1985

24. NASA Lewis - Cleveland, OH, "Optical Linear Algebra Processors (Systolic)".

March 1985

25. George Washington University, - Washington, D.C., "Optical Linear Algebra for SDI".
26. Lockheed Missiles & Space Co. - Sunnyvale, CA, "Advanced Hybrid Optical/Digital Pattern Recognition"
27. OSA Topical Meeting on Optical Computing - Lake Tahoe, NV, "Fabrication and Testing of a Space and Frequency-Multiplexed Optical Linear Algebra Processor".
28. OSA Topical Meeting on Machine Vision - Lake Tahoe, NV, "Hierarchical Feature-Based Object Identification".
29. OSA Topical Meeting on Machine Vision - Lake Tahoe, NV, "Correlation Filters for Distortion-Invariance and Discrimination".
30. Texas Instruments - Dallas, TX, "Optical Pattern Recognition".

April 1985

31. Electro-Com Automation, Inc. - Dallas, TX, "Optical Pattern Recognition".
32. Eglin Air Force Base - Ft. Walton Beach, FL, "Optical Pattern Recognition and Kalman Filtering".

May 1985

33. Carnegie-Mellon University - Board of Trustees, "Optical Data Processing".

August 1985

34. SPIE - San Diego, CA, "Correlation Synthetic Discriminant Functions for Object Recognition and Classification in High Clutter".
35. SPIE - San Diego, CA, "A Factorized Extended Kalman Filter".

September 1985

36. SPIE - Cambridge, MA, "Parameter Estimation and In-Plane Distortion Invariant Chord Processing".
37. SPIE - Cambridge, MA, "Optical Processing Techniques for Advanced Intelligent Robots and Computer Vision".
38. SPIE - Cambridge, MA, "High-Dimensionality Feature-Space Processing with Computer Generated Holograms".

October 1985

39. SDI - Washington, D.C., "Optical Data Processing for SDI".
40. Martin Marietta - Denver, CO, "Optical Data Processing".

November 1985

41. IEEE Computer Society, *Workshop on Computer Architectures for Pattern Analysis and Image Database Management* - Miami Beach, FL, "Optical Computer Architectures for Pattern Analysis".

January 1986

42. SPIE Engineering Update Series, "Fourier Optics for Electrical Engineers" - Los Angeles, CA.
43. SPIE Engineering Update Series, "Optical Data Processing", Los Angeles, CA.
44. SPIE Conference - Los Angeles, CA, "A Feature Space Rule-Based Optical Relational Graph Processor".
45. SPIE Conference - Los Angeles, CA, "Optical Linear Algebra Processors: Architectures and Algorithms".
46. SPIE Conference - Los Angeles, CA, "Optical AI Symbolic Correlators: Architecture and Filter Considerations".
47. Optical Society of America - Los Angeles, CA, "Optical Computing".
48. Corporate Advisory Group on Optical Information Processing - Los Angeles, CA, "Optical Computing".
49. Jet Propulsion Laboratory/NASA - Pasadena, CA, "Optical Linear Algebra and Pattern Recognition Processors".

February 1986

50. Computer Science Department, Carnegie-Mellon University - Pittsburgh, PA, "Optical AI Pattern Recognition Research in ECE".

March 1986

51. Carnegie-Mellon University, Professional Education Program - Pittsburgh, Pennsylvania, "Optical Data Processing".
52. Air Force Institute Conference of Technology - Dayton, Ohio, "Optical Data Processing at Carnegie-Mellon University".
53. Mars Electronics - Philadelphia, PA, "Optical Pattern Recognition".
54. SPIE Advanced Institute Series on Hybrid and Optical Computers - Leesburg,

Virginia, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence".

April 1986

55. SPIE Conference - Orlando, FL, "Model-Based System for On-Line Affine Image Transformations".
56. Robotics Institute - Carnegie-Mellon University - Pittsburgh, PA, "Optical AI Pattern Recognition Research in ECE".

May 1986

57. IBM, Federal Systems Division - Manassas, VA, "Optical Computing".
58. General Electric - Philadelphia, PA, "Adaptive Optical Processing".
59. Litton Data Systems - Van Nuys, CA, "Multiple Degree of Freedom Pattern Recognition".
60. Rockwell Corporation - Seal Beach, CA, "Optical Signal Processing".
61. NASA Jet Propulsion Laboratory, California Institute of Technology - Pasadena, CA, "Multiple Degree of Freedom Optical Pattern Recognition".
62. SPIE Engineering Update Series, "Fourier Optics and Components for Electrical Engineers" - Los Angeles, CA.
63. Philip Morris Corporation - Richmond, VA, "Applications of Optical Data Processing to Automated Inspection".

June 1986

64. Carnegie-Mellon University, Professional Education Program - Pittsburgh, PA, "Optical Pattern Recognition".
65. Carnegie-Mellon University, Professional Education Program - Pittsburgh, PA, "Optical Signal Processing".
66. SPIE Engineering Update Series, "Fourier Optics and Components for Electrical Engineers" - Tufts University, Boston, MA. Boston, MA - "Operations Achievable".
67. University of Pretoria - Pretoria, South Africa, "Optical Data Processing".

July 1986

68. IOCC Conference - Jerusalem, Israel, "Optical Artificial Intelligence Processors".

August 1986

69. SPIE Conference - San Diego, CA, "Distortion-Invariant Associative Processors".

September 1986

70. ALCOA - Pittsburgh, PA, "Optical Information Processing".
71. General Electric - Philadelphia, PA, "Optical Processing".
72. Eikonix Corp. - Boston, MA, "Optical Pattern Recognition for Optical Character Recognition".
73. Penn State University - State College, PA, "Optical Scene Analysis and Artificial Intelligence".

October 1986

74. Advanced Technology Intl. - Boston, MA, "Optical Information Processing".
75. Advanced Technology Intl. - Orlando, FL, "Optical Information Processing".
76. Advanced Technology Intl. - Washington, D.C., "Optical Information Processing".
77. Carnegie-Mellon University, Professional Education Program (presented to IBM) - Pittsburgh, Pennsylvania, "Optical Pattern Recognition".
78. Carnegie-Mellon University, Professional Education Program (presented to IBM) - Pittsburgh, Pennsylvania, "Optical Data Processing".

79. SPIE Conference - Boston, MA, "Hierarchical Processor and Matched Filters for Range Image Processing".
80. SPIE Conference - Boston, MA, "Large Class Iconic Pattern Recognition: An OCR Case Study".
81. Carnegie Mellon University, ECE Graduate Seminar - Pittsburgh, PA, "Optical Computing in ECE: 1986".

November 1986

82. ICALEO'86 - Arlington, VA, "Advanced Optical Pattern Recognition and Artificial Intelligence".
83. Optical Society of America (San Diego Chapter) - San Diego, CA, "Optical Computing".

December 1986

84. Philip Morris - Richmond, VA, "Optical Pattern Recognition for Inspection and Robotics".
85. ORD - Washington, D.C., "Optical Computing Accomplishments".

January 1987

86. SPIE Conference - Los Angeles, CA, "A Directed Graph Optical Processor".
87. SPIE Conference - Los Angeles, CA, "Complex Data Handling in Analog and High-Accuracy Optical Linear Algebra Processors".
88. SPIE Conference - Los Angeles, CA, "Parameter Selection for Iconic and Symbolic Pattern Recognition Filters".
89. SPIE Conference - Los Angeles, CA, "1-D Acousto Optic Processing of 2-D Image Data".
90. SPIE Conference - Los Angeles, CA, "Optical Pattern Recognition and Artificial Intelligence: A Review" (Invited Keynote Speaker).
91. SPIE Conference - Los Angeles, CA, "Optical Pattern Recognition and AI Algorithms and Architectures for ATR and Computer Vision" (Invited).
92. SPIE Conference - Los Angeles, CA, "Electro Optic Target Detection and Object Recognition" (Invited).
93. Workshop on Space Telerobotics - NASA JPL, Pasadena, CA, "Multiple Degree of Freedom Optical Pattern Recognition".
94. Hewlett Packard - Palo Alto, CA, "Optical Computing".

February 1987

95. ISC Defense Systems, Inc. - Lancaster, PA, "Optical Computing and Signal Processing".
96. DARPA - Washington, D.C., "Optical Computing: A Review"

March 1987

97. Advanced Technology Intl., Short Course - Los Angeles, CA, "Optical Information Processing".
98. Advanced Technology Intl., Short Course - San Diego, CA, "Optical Information Processing".
99. Advanced Technology Intl., Short Course - Anaheim, CA, "Optical Information Processing".
100. Advanced Technology Intl., Short Course - Palo Alto, CA, "Optical Information Processing".
101. Aerospace Corporation - Los Angeles, CA, "Optical Computing and Signal Processing Research at CMU".

102. OSA Topical Meeting on Optical Computing - Lake Tahoe, NV, "Rule-Based, Probabilistic, Symbolic Target Classification by Object Segmentation".

May 1987

103. NASA Langley Research Center - Hampton, VA, "Machine Vision".

June 1987

104. Perkin-Elmer - White Plains, NY, "Optical Computing".

July 1987

105. Carnegie Mellon University - ECE Department, Presentation to the attendees of the Fault Tolerant Computing Conference, Pittsburgh, PA.

August 1987

106. UCLA Extension Course - Los Angeles, CA, "Optical Computing".
107. Mathematical Modeling Conference - St. Louis, MO, "Computations with Optical Computers".
108. TRW - Los Angeles, CA, "Optical Data Processing of Synthetic Aperture Radar Signals for Pattern Recognition".
109. Galileo - Sturbridge, MA, "Product Opportunities in Optical Data Processing".
110. General Electric - Valley Forge, PA, "Recent Progress in Adaptive Optical Data Processing".

September 1987

111. Defense Science Board, Pentagon - Washington, D.C, "Optical Computing for Automatic Target Recognition".

October 1987

112. AIAA Computers in Aerospace VI Conference - Boston, MA, "Multi-Functional Optical Logic, Numerical and Pattern Recognition Processor".
113. Philip Morris Corporation - Richmond, VA, "Optical Processing for Product Inspection".

November 1987

114. SPIE Robotics Conference - Boston, MA, "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results".
115. SPIE Robotics Conference - Boston, MA, "Rule-Based String Code Processor".
116. SPIE Robotics Conference - Boston, MA, "Model-Based Satellite Acquisition and Tracking".
117. SPIE Robotics Conference - Boston, MA, "Optical Processor for Product Inspection".
118. SPIE Robotics Conference - Boston, MA, "Optical Feature Extraction for High-Speed Inspection".
119. SPIE Robotics Conference - Boston, MA, "Multi-Sensor Processing: Object Detection and Identification"

December 1987

120. National Security Agency - Maryland, "Optical Information Processing".

12.3 THESES SUPPORTED BY AFOSR FUNDING (SEPTEMBER 1984-DATE)

1. Eugene Pochapsky, M.S. Dissertation, "The Simulation of Optical Pattern Recognition Systems", September 1984.

2. William Rozzi, M.S. Dissertation, "Advanced Quantitative Synthetic Discriminant Function Tests on Ship Imagery", December 1984.

3. James Fisher, M.S. Dissertation, "Extended Kalman Filter Algorithms for Implementation on a High-Accuracy Optical Processor", December 1984.

4. W.T. Chang, Ph.D. Dissertation, "Chord Distributions and Correlation SDFs in Pattern Recognition", March 1985.

5. Andrew J. Lee, M.S. Dissertation, "High-Dimensionality Feature Space Pattern Recognition Using Computer Generated Holograms", January 1986.

6. Abhijit Mahalanobis, M.S. Dissertation, "Application of Synthetic Discriminant Functions for Optical Character Recognition", September 1985.

7. Jeffrey Richards, M.S. Dissertation, "Optical Processing for Product Inspection", November 1986.

8. Brian Telfer, M.S. Dissertation, "Optical Associative Memories for Distortion-Invariant Pattern Recognition", February 1987.

9. Abhijit Mahalanobis, Ph.D. Dissertation, "New Correlation Filters for Symbolic Rule-Based Pattern Recognition", August 1987.

10. Raghuram Krishnapuram, Ph.D. Dissertation, "Hough Space Associative Processor for Pattern Recognition", August 1987.

FILMED

7 88